CRT-2000-34

Centre de recherche sur les transports (C.R.T.)
Centre for Research on Transportation

PROCÉDURE DE DIVERSIFICATION POUR LA RÉSOLUTION DES PROBLÈMES STOCHASTIQUES DE TOURNÉES DE VÉHICULES PAR L'ALGORITHME TABOU

par

Joséphine-Muriel Pelleu-Tchétagni

Mémoire d'étudiant

Novembre 2000

101-01698/

PUBLICATION
CRT-2000-34

PROCÉDURE DE DIVERSIFICATION POUR LA RÉSOLUTION DES PROBLÈMES STOCHASTIQUES DE TOURNÉES DE VÉHICULES PAR L'ALGORITHME TABOU*

par

Joséphine-Muriel Pelleu-Tchétagni[1]

* Mémoire présenté en vue de l'obtention du diplôme de Maîtrise ès sciences. Sous la direction de Michel Gendreau et co-direction de Patrick Soriano. Ce mémoire a été rendu possible grâce en partie au Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG) et aux Fonds pour la Formation de Chercheurs et l'Aide à la Recherche (FCAR).

[1] Département d'informatique et de recherche opérationnelle et Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7

Département d'informatique et de recherche opérationnelle - Publication #1183

© Joséphine-Muriel Pelleu-Tchétagni, 2000

Dépôt légal – Bibliothèque nationale du Québec
Bibliothèque nationale du Canada, 2000

Remerciements

Ce mémoire a pu être complété grâce à l'aide et aux conseils de mon directeur Michel Gendreau, et de mon codirecteur Patrick Soriano. Je leur témoigne une grande reconnaissance pour leur patience et leur disponibilité. Je remercie également René Séguin, qui m'a apporté de nombreuses recommandations techniques dans mon travail. Je tiens enfin à exprimer toute ma gratitude à ma famille, mes amis, et surtout mes parents pour leur soutien moral au cours des cinq dernières années.

Par ailleurs, je présente mes compliments au Centre de Recherche sur les Transports, qui encourage l'esprit de recherche et d'efficacité, en mettant à la disposition des étudiants un matériel performant, ainsi qu'un personnel disponible et serviable.

Sommaire

Mots clefs

Tournées stochastiques, Métaheuristique, Recherche tabou, Diversification, Cross-exchange

Les problèmes stochastiques de tournées de véhicules sont des cas particuliers avec composantes stochastiques du problème de voyageur de commerce et du problème de tournées de véhicule. Plus représentatifs de la réalité que leur variante déterministe, ils font l'objet d'un intérêt croissant en optimisation combinatoire. Cette classe de problèmes peut être partitionnée en trois groupes et dans chacun d'eux, un des paramètres du problème déterministe est une variable aléatoire. Ainsi, on aura le problème de tournées de véhicules avec demandes stochastiques (PTVDS), le problème de voyageur de commerce avec clients stochastiques (PVCCS) et le problème de tournées de véhicules avec temps de services stochastiques. De ces trois catégories, on peut également inférer des variantes où, plusieurs de ces paramètres peuvent être stochastiques en même temps. De nombreuses heuristiques ont été développées pour les résoudre. Parmi elles, la métaheuristique de recherche tabou a été mise à contribution avec succès. Par exemple, les travaux dus à Séguin[60], présentent une très bonne performance d'une recherche tabou simple dans la résolution d'un problème de tournées de véhicules avec demandes et clients stochastiques (PTVDCS). Cependant, des erreurs pouvant aller jusqu'à 20% demeuraient. Nous présentons dans ce

mémoire une procédure de diversification par *cross-exchange* ajoutée à cet algorithme tabou. Nous montrons qu'elle permet d'éliminer ou de réduire significativement la plupart des erreurs observées dans l'algorithme de base. Nous proposons finalement d'autres expériences supplémentaires qui pourraient être envisagées, afin d'améliorer la performance de la diversification. Les expérimentations ont été effectuées pour le PVCCS, le PTVDS et le PTVDCS.

Table des matières

Remerciements i

Sommaire ii

Table des matières iv

Liste des tableaux vii

Liste des figures xi

Liste des annexes xii

Introduction 1

1 La méthode de recherche tabou 8

1.1 Principes de base: la mémoire à court terme 9

1.1.1 Exemple: Le problème d'ordonnancement des tâches 10

1.1.2 Les listes tabous 11

1.1.3 Le critère d'aspiration 13

1.1.4 L'algorithme tabou 13

1.2 Éléments additionnels: la mémoire à long terme 14

1.2.1 La diversification et l'intensification 15

1.2.2 L'oscillation stratégique 17

1.3 Raffinements 17

1.3.1 Les listes tabous 18
1.3.2 L'exploration du voisinage 18

2 La recherche tabou appliquée aux problèmes stochastiques de tournées de véhicules 20
2.1 Définition des problèmes 20
2.1.1 Le problème du voyageur de commerce avec clients stochastiques (PVCCS) 20
2.1.2 Le problème de tournées de véhicules avec demandes stochastiques (PTVDS) 22
2.1.3 Le problème de tournées de véhicules avec clients et demandes stochastiques (PTVDCS) 23
2.2 Application de la recherche tabou aux problèmes stochastiques de tournées de véhicules 24
2.3 Application aux autres problèmes de tournées de véhicules 27

3 La procédure de diversification 28
3.1 Approche générale: le cross-exchange 28
3.1.1 Variantes du cross-exchange 30
3.1.2 Mécanisme et précautions 33
3.2 Structures de données 36
3.3 Choix algorithmiques 37
3.3.1 Choix du nombre de sommets dans les séquences 37
3.3.2 Choix des séquences 38
3.3.3 Le déplacement des séquences 40
3.4 Mise en oeuvre 40
3.4.1 Inter-routes 42
3.4.2 Intra-route 43
3.5 L'algorithme tabou avec diversification 44

4 Résultats **48**
4.1 Types de problèmes 49
4.1.1 Le PTVDCS 49
4.1.2 Le PVCCS 50
4.1.3 Le PTVDS 50
4.2 Critères d'arrêt 51
4.3 Performance de l'algorithme tabou avec diversification 53
4.3.1 Le PTVCDS 54
4.3.2 Le PVCCS 59
4.3.3 Le PTVDS 62
4.4 Analyse de la performance de l'algorithme tabou avec diversification 64
4.5 Expérimentations supplémentaires 67
4.6 Travaux futurs 71
4.6.1 Expérimentations avec plus d'essais dans la diversification . . 71

Conclusion **75**

Liste des tableaux

2.1 Les problèmes stochastiques de tournées de v hicules. 24

4.1 Types de problèmes testés dans le PTVCDS lorsqu'il y a n clients . . 49
4.2 Types de problèmes testés dans le PVCCS lorsqu'il y a n clients . . . 51
4.3 Types de problèmes testés dans le PTVDS 51
4.4 Critères d'arrêt pour les deux heuristiques PTVCDS 51
4.5 Critères d'arrêt pour les deux heuristiques dans le PVCCS 52
4.6 Critères d'arrêt pour les deux heuristiques dans le PTVDS 52
4.7 Erreurs maximales globales du tabou et erreurs correspondantes du tabou+diversification dans le PTVCDS 54
4.8 Erreurs maximales globales du tabou+diversification et erreurs correspondantes du tabou dans le PTVCDS 55
4.9 Performance des deux algorithmes en fonction de s dans le PTVCDS 58
4.10 Erreurs par intervalle pour les deux algorithmes dans le PTVCDS . . 58
4.11 Erreurs maximales globales du tabou et erreurs correspondantes du tabou+diversification dans le PVCCS 59
4.12 Erreurs maximales globales du tabou+diversification et erreurs correspondantes du tabou dans le PVCCS 60
4.13 Performance des deux algorithmes en fonction de $[a, b]$ dans le PVCCS 61
4.14 Erreurs par intervalle pour les deux algorithmes dans le PVCCS . . . 62

4.15 Erreurs maximales globales du tabou et erreurs correspondantes du tabou+diversification dans le PTVDS 63
4.16 Erreurs maximales globales du tabou+diversification et erreurs correspondantes du tabou dans le PTVDS 63
4.17 Performance des deux algorithmes en fonction de f dans le PTVDS . 64
4.18 Erreurs par intervalle pour les deux algorithmes dans le PTVDS . . . 64
4.19 Erreurs moyennes des deux algorithmes selon le type de problèmes dans le PVCCS 65
4.20 Performance des deux algorithmes en fonction de f dans le PTVCDS 66
4.21 Critères d'arrêts pour les expérimentations supplémentaires(n clients) 68
4.22 Caractéristiques des problèmes de plus grande taille 68
4.23 Erreurs moyennes [*#erreurs*] totales pour les 5 critères d'arrêts 69
4.24 Erreurs moyennes [*#erreurs*] totales en fonction de *(s, f)* pour les 5 critères d'arrêts 70
4.25 Caractéristiques des problèmes où plus d'une route est choisie aléatoirement 71
4.26 Impact du nombre de routes choisies aléatoirement sur la performance de la diversification: *erreur moyenne par taille* [*# erreurs*] en fonction de (s, f) . 72
4.27 Impact du nombre de routes choisies aléatoirement sur la performance de la diversification: *erreur moyenne par taille* [*# erreurs*] en fonction de s 73

A.1 Erreur maximale pour TD1 dans le PTVCDS 88
A.2 Erreur maximale pour T2 dans le PTVCDS 89
A.3 Erreur maximale pour TD2 dans le PTVCDS 90
A.4 Erreur maximale pour T4 dans le PTVCDS 91
A.5 Erreur maximale pour TD4 dans le PTVCDS 92
A.6 Erreur maximale pour T8 dans le PTVCDS 93
A.7 Erreur maximale pour TD8 dans le PTVCDS 94
A.8 Erreur maximale pour T16 dans le PTVCDS 95

A.9 Erreur moyenne et nombre d'erreurs pour TD1 dans le PTVCDS . . 96
A.10 Erreur moyenne et nombre d'erreurs pour T2 dans le PTVCDS . . . 97
A.11 Erreur moyenne et nombre d'erreurs pour TD2 dans le PTVCDS . . 98
A.12 Erreur moyenne et nombre d'erreurs pour T4 dans le PTVCDS . . . 99
A.13 Erreur moyenne et nombre d'erreurs pour TD4 dans le PTVCDS . . 100
A.14 Erreur moyenne et nombre d'erreurs pour T8 dans le PTVCDS . . . 101
A.15 Erreur moyenne et nombre d'erreurs pour TD8 dans le PTVCDS . . 102
A.16 Erreur moyenne et nombre d'erreurs pour T16 dans le PTVCDS . . 103

B.1 Erreur maximale pour TD4 dans le PVCCS 105
B.2 Erreur maximale pour T8 dans le PVCCS 105
B.3 Erreur maximale pour TD8 dans le PVCCS 106
B.4 Erreur maximale pour T16 dans le PVCCS 106
B.5 Erreur moyenne[#d'erreurs] pour TD4 dans le PVCCS 107
B.6 Erreur moyenne[#d'erreurs] pour T8 dans le PVCCS 107
B.7 Erreur moyenne[#d'erreurs] pour TD8 dans le PVCCS 108
B.8 Erreur moyenne[#d'erreurs] pour T16 dans le PVCCS 108
B.9 Erreur maximale de l'algorithme tabou+diversification selon le type de problèmes dans le PVCCS 109
B.10 Erreur maximale de l'algorithme tabou selon le type de problèmes dans le PVCCS 109
B.11 Erreur moyenne de l'algorithme tabou+diversification selon le type de problèmes dans le PVCCS 109
B.12 Erreur moyenne de l'algorithme tabou selon le type de problèmes dans le PVCCS 109

C.1 Erreur maximale pour TD4 dans le PTVDS 110
C.2 Erreur maximale pour T8 dans le PTVDS 111
C.3 Erreur moyenne[#d'erreurs] pour TD4 dans le PTVDS 111

C.4 Erreur moyenne[#d'erreurs] pour T8 dans le PTVDS 112

D.1 Erreurs[#erreurs] de l'algorithme tabou+diversification relativement à la meilleure solution connue: DIV1 114

D.2 Erreurs[#erreurs] de l'algorithme tabou+diversification relativement à la meilleure solution connue: DIV2 114

D.3 Erreurs[#erreurs] de l'algorithme tabou+diversification relativement à la meilleure solution connue: DIV3 115

D.4 Erreurs[#erreurs] de l'algorithme tabou+diversification relativement à la meilleure solution connue: DIV4 115

D.5 Erreurs[#erreurs] de l'algorithme tabou+diversification relativement à la meilleure solution connue: TAB 116

Liste des figures

2.1 Instance d'un tour dans le PVCCS 21
2.2 Rupture au client 7 dans le PTVDS 22

3.1 Cross-exchange à l'intérieur d'une route: Intra-route 29
3.2 Cross-exchange entre 2 routes distinctes: Inter-routes 29
3.3 Cas général du cross-exchange: plusieurs paires de segments sont impliquées 30
3.4 Cas particulier du cross-exchange: une seule séquence change de route 32
3.5 Sens de parcours des segments interchangés dans l'intra-route 33
3.6 Sens de parcours à ignorer dans l'intra-route 34
3.7 Sens de parcours dans l'inter-routes 35
3.8 Algorithme de diversification 41
3.9 Algorithme tabou avec diversification 45

Liste des annexes

A Erreurs maximales et erreurs moyennes par taille dans le PTVDCS 87

B Erreurs maximales et erreurs moyennes par taille dans le PVCCS 104

C Erreurs maximales et erreurs moyennes par taille dans le PTVDS 110

D Résultats détaillés de la section 4.5: erreurs moyennes par taille . 113

Introduction

La localisation géographique des biens et services est un aspect de première importance pour un rendement optimal dans leur exploitation. Pour atteindre ce but, deux modèles de base en optimisation combinatoire ont été développés. Il s'agit du problème du voyageur de commerce (PVC) et du problème de tournées de véhicules (PTV).

Dans un PVC de taille n, un véhicule de capacité illimitée partant d'un dépôt doit effectuer un trajet où n points (clients) doivent être visités chacun exactement une fois. À chaque point $i = 1, 2, \ldots, n$, une quantité d_i doit être collectée ou déposée. Le véhicule doit retourner à son point de départ après avoir visité tous les points. Le but du PVC est de déterminer un ordre de parcours (tour) qui minimise la valeur d'une fonction objectif f. Celle-ci peut être la distance totale de parcours, le temps total de parcours ou un autre paramètre.

Le PTV peut être vu comme une généralisation du PVC. En effet, le processus correspondant est similaire: plusieurs points doivent être visités exactement une fois. Néanmoins, dans un tel problème de taille n, les n points seront répartis dans plusieurs tours plutôt qu'un seul. Chaque tour est parcouru par un véhicule unique. Les véhicules sont de capacités identiques mais pas infinies. Ils commencent et achèvent leur trajet à un même point de départ: le dépôt. Les demandes d_i à chaque point ne sont plus identiques. Le but est de minimiser la valeur d'une fonction objectif f définie sur l'ensemble de toutes les tournées. Lorsque d_i est non nulle, une contrainte de capacité est appliquée à chaque véhicule. On obtient le problème de tournées de

véhicules avec contrainte de capacité (PTVC). En plus de parcourir chaque point, les véhicules devront maintenant satisfaire la demande correspondante. Dans le PTVC, on se propose de satisfaire la demande des clients dans chaque tournée, sans excéder la capacité du véhicule impliqué, tout en minimisant la valeur de f. Par ailleurs, quand le temps disponible pour servir les clients est sujet à une borne supérieure, on a un PTV avec des tournées dont le temps de parcours a une durée limitée. Ce problème donne lieu au PTV avec fenêtres de temps.

Comme le témoignent certaines études publiées, de nombreuses situations réelles peuvent être représentées par chacun de ces deux problèmes. Dans l'article de Basnet et al. [6], le problème de distribution de produits laitiers est résolu comme un PTV. Dans Braca et al. [11] et Trudeau et al. [70] respectivement, les problèmes du trajet des bus scolaires et des ambulances sont traités. D'autres applications comme les collectes d'argent aux guichets automatiques (Lambert et al. [36]) ou encore le déneigement des communes (Eglese [16]), peuvent être transformées dans un PTV.

Le problème de voyageur de commerce quant à lui, peut être utilisé dans des situations telles que la distribution du courrier (Letchford et al. [42]). Dans les travaux de Kwan et al. [33] ou de Bartholdi et al. [5], on trouve d'autres débouchés pratiques de ces problèmes.

Le premier modèle du PVC a été développé dans Dantzig et al. [14]. Une extension plus générale a suivi pour le PTV dans Dantzig et Ramser [13]. Depuis lors, plusieurs formes analogues, dérivées de ces modèles de base, ont été résolues à l'aide d'algorithmes exacts.

Dans l'article de Bard et al. [4], une méthode de plans de coupures (*"Branch and Cut"*) permet de résoudre un PTV avec services satellites. Dans Pesant et al. [55], l'approche de séparation et évaluation progressive (*"Branch and Bound"*) est utilisée dans un PVC avec fenêtres de temps. Néanmoins, bien que ces méthodes garantissent de trouver une solution optimale globale au problème, les efforts de calculs qu'elles requièrent sont généralement considérables.

Pour supprimer cette difficulté, des techniques permettant de trouver des solutions de très bonne qualité dans un intervalle de temps raisonnable ont été développées. Cependant, l'obtention d'une solution optimale globale au problème n'est plus garantie. Ces méthodes communément appelées *heuristiques*, se basent sur l'exploration de l'ensemble des solutions possibles du problème. Dans cette recherche, l'algorithme part d'une solution initiale donnée et *visite* tour à tour d'autres solutions possibles du problème. On génère une autre solution en effectuant une ou plusieurs *transformations* sur la solution courante.

Dans l'optimisation du PVC et du PTV, les heuristiques les plus souvent utilisées sont les *méthodes de descente*. À l'itération l d'une méthode de descente, l'algorithme passe d'une solution s_l à une solution s_{l+1} seulement si la valeur de la fonction objectif pour cette dernière est plus petite que celle de s_l. Un grand nombre de publications présentent des méthodes de descente pour résoudre les problèmes de transport. Leur performance dépend en majeure partie des transformations appliquées aux solutions connues pour en générer d'autres (définition du voisinage). Dans Or [50] et Lin et al. [43] respectivement, les heuristiques d'échanges *Or-Opt* et *k-Opt* sont définies et utilisées pour l'application du PVC dans des situations réelles. L'heuristique d'échange de segments de tour (arêtes) *k-exchange* a été utilisée dans Psarafatis [56] pour la résolution d'un PTV avec contraintes de précédence entre les points. Pour cette même famille de problèmes, l'approche *3-Opt* est appliquée dans Alfa et al. [1] pour la résolution d'un PTV classique. Une revue plus détaillée de ces travaux est présentée dans Laporte et Osman [39].

Malgré leur grande utilisation, les méthodes de descente locale ont un défaut majeur. En effet, ces algorithmes évoluent seulement quand la solution générée se trouve être la meilleure jamais trouvée. De plus, les transformations utilisées dans la recherche ne permettent pas toujours de générer toutes les solutions possibles. Il s'ensuit donc une exploration régionale (locale) de l'espace des solutions. Cela implique que la solution finale de la recherche n'est pas nécessairement un optimum global. Pour attribuer

plus de flexibilité à la recherche, l'idée est de la diriger de manière plus intelligente, plutôt que de la baser sur une seule règle stricte. Cette optique nouvelle a ouvert l'ère des *métaheuristiques*. La particularité de ces méthodes est qu'elles guident les simples heuristiques de manière à vaincre l'optimalité locale. Les principes d'*adaptation* et de *diversité* sont généralement la clé de ce contrôle. En effet, les métaheuristiques sont presque toujours caractérisées par deux processus interdépendants qui collaborent. Le premier processus est en général une heuristique simple qui est par exemple une descente locale (Mladenovic et Hansen [48]). Ce dernier sera dirigé ou influencé par un deuxième mécanisme qui lui imposera différentes règles dans son évolution.

Par exemple, dans le *recuit simulé* (Osman [53]), on utilise un paramètre appelé température afin de contrôler l'acceptation des transformations d'une solution courante qui résultent dans une dégradation de la fonction objectif. Les fluctuations appliquées à ce paramètre permettront au processus de recherche de se déplacer vers des régions variées.

Dans les *algorithmes génétiques* (Mitchell et al. [47]), la recherche est effectuée sur le voisinage d'un ensemble de solutions plutôt que d'une seule. De plus, on applique de manière périodique des transformations sur la solution courante qui s'apparentent aux véritables phénomènes d'évolution génétique. Le but est de générer des solutions variées par *mutation* ou *croisement* ou encore, des solutions de bonne qualité par *sélection*.

La *recherche tabou* (RT) quant à elle est basée sur une recherche locale dirigée par des règles qui ont pour but de l'empêcher de cycler à court terme, c'est-à-dire de retomber sur des solutions déjà examinées dans un passé proche. Pour cause, la recherche locale n'est plus descente stricte où, étant donné que seules les solutions qui améliorent la fonction objectif seraient acceptées, il est certain qu'une solution ne sera jamais acceptée plus d'une fois. Ici, la recherche locale se contente d'examiner le plus de solutions possibles. Par conséquent, s'il n'y a pas de restrictions sur l'exploration d'un voisinage, il n'est pas exclu de retomber sur des solutions déjà examinées. Les

règles que doit respecter une RT sont tirées de l'information du passé de la recherche. Ces règles sont matérialisées à travers des *listes tabou* considérées comme mémoire à court terme. On trouve dans Glover [27] une description de base des principes de la RT. La bibliographie de Laporte et Osman [54] en procure une présentation plus détaillée. Il s'y trouve également de nombreuses références sur le sujet.

Bien que le PTV et le PVC aient été considérablement étudiés, il demeure des situations réelles où certaines de leurs composantes sont imprévisibles. En effet, la présence, le niveau de demande ou le temps de service des clients peuvent être sujets à des aléas. Cette question a donné lieu à l'étude des variantes stochastiques des problèmes de transports. Plusieurs modèles ont été proposés pour définir ces problèmes. Ils se basent généralement sur le concept de *programmation stochastique* (Dror et al. [15]) ou d'*optimisation a priori* (Jaillet [32]). Ces problèmes étant difficiles, ils ont suscité un intérêt notoire dans le domaine de l'optimisation combinatoire. On peut trouver une bibliographie complète des articles sur ce sujet dans Gendreau et al. [22]. À notre connaissance, une des rares études du problème stochastique de transport avec plus d'une composante stochastique se trouve dans Séguin [60]. Dans ce travail, le PTV avec clients et demandes stochastiques (PTVCDS) est résolu exactement par la méthode de séparation et évaluation progressive. Ensuite, une heuristique tabou avec intensification est utilisée pour obtenir un effort de calcul plus performant. La confrontation des résultats des solutions optimales connues et de l'algorithme tabou a démontré une robustesse significative de ce dernier. Néanmoins, des erreurs allant jusqu'à 20% demeuraient. Cette constatation montre que la RT simple peut avoir des limitations majeures dans certains cas, malgré sa supériorité sur les méthodes classiques de descente. Dans le cadre de l'étude dans [60], nous savons qu'il existe une meilleure solution x^*. L'incapacité de la RT à trouver cette solution pourrait provenir du fait que les opérateurs du voisinage n'arrivent pas à diriger la recherche vers x^* ou son voisinage. Il se pourrait aussi que les listes tabous ne soient plus efficaces à long terme. En effet, leur longueur limitée empêche de retomber sur des

solutions déjà visitées seulement à court terme mais pas indéfiniment. Il n'est donc pas exclu que la RT cycle à long terme.

Selon Soriano et Gendreau dans [63], l'introduction d'une stratégie de diversification dans une RT simple pourrait pallier à ces problèmes. En effet, le but de la diversification est de *forcer* la recherche vers des *nouvelles* régions de l'espace des solutions, c'est-à-dire des régions peu ou pas explorées dans le passé.

Une procédure de diversification peut être introduite dans une RT simple de diverses façons. On pourrait relancer la recherche à partir d'une nouvelle solution. Cette solution est sélectionnée en se basant sur des informations du passé lointain (ou à long terme) de la recherche. Par exemple, on pourrait établir une mémoire contenant les fréquences des mouvements effectués ou encore, les fréquences de présence de certains attributs dans les solutions. Le choix des nouvelles solutions de départ se baserait alors sur une telle mémoire. Cette méthode est appliquée dans Laporte et al.[40] pour la résolution d'un PTV avec groupement des sommets.

La diversification pourrait aussi intervenir de manière indirecte ou continue, sans l'interruption de la RT de base. Dans ce cas, en utilisant les informations de la mémoire à long terme, on pourrait introduire un terme de pénalité dans la fonction objectif. Ce terme pénaliserait l'exécution des mouvements fréquemment effectués ou encore, l'inclusion dans une solution des attributs fréquemment utilisés. Une approche se basant sur cette idée est utilisée dans Berger [7] pour la résolution d'un problème de conception de réseaux.

Dans ce mémoire, notre objectif est *d'analyser l'impact qu'apporterait l'introduction d'une stratégie de diversification dans la performance de la RT simple étudiée dans [60], notamment, sur les grosses erreurs qui demeurent.* En plus du PTVCDS, nous étudions également les résultats pour le problème de tournées de véhicules avec demandes stochastiques (PTVDS) et pour le problème de voyageur de commerce avec clients stochastiques (PVCCS). La raison de l'étude de plusieurs variantes stochastiques du PTV est de pouvoir tirer des conclusions relativement générales quant à

l'utilité de la diversification dans le contexte de la RT appliquée dans la résolution de ces problèmes.

Les solutions des PTV correspondent à des routes définissant un ordre de parcours des clients qu'elles contiennent. Étant donné que ces clients ont des caractéristiques stochastiques, il est raisonnable de penser qu'un changement de leur ordre de parcours dans une solution donnée pourrait engendrer une solution *très différente*. Pour cette raison, nous avons choisi dans ce contexte particulier d'appliquer l'opérateur *Cross-exchange* à une solution donnée en guise de diversification. La RT sera relancée à partir de la solution résultante.

La méthode exacte développée dans [60] sera donc aussi appliquée au PTVDS et au PVCCS. Ensuite, en se basant sur les meilleures solutions connues, nous comparerons la performance de la RT simple et celle de la RT avec diversification pour les trois types problèmes.

L'analyse de la performance de l'algorithme tabou avec diversification sera effectuée en fonction du nombre de clients stochastiques et du rapport de la demande totale espérée des clients et de la capacité des véhicules. Cela permettra de qualifier l'impact de la diversification en fonction des valeurs des composantes stochastiques des problèmes.

Le reste du document est organisé en quatre chapitres. Dans le chapitre 1, les principes de base de la recherche tabou sont présentés. Au chapitre 2, nous définissons les trois types de problèmes de transports stochastiques auxquels l'algorithme tabou avec diversification sera appliqué. Le chapitre 3 définit la procédure de diversification employée à cette fin. Enfin, les caractéristiques des expérimentations numériques seront définies au chapitre 4. Nous y présenterons aussi les résultats des deux heuristiques. En nous basant sur ces résultats, nous effectuerons une analyse paramétrique de l'amélioration de la performance de la RT, due à la diversification. Cela nous permettra de tirer une conclusion relativement fondée quant à l'utilité d'une stratégie de diversification dans la résolution des ces problèmes par la RT.

Chapitre 1

La méthode de recherche tabou

Les méthodes traditionnelles de descente sont telles qu'il est fréquent que l'algorithme s'achève sur un optimum local. Le but de la recherche tabou (RT) est de pallier à cette faiblesse par l'utilisation de mécanismes plus sophistiqués pour diriger l'algorithme de manière intelligente. Pour ce faire, la recherche tabou exploite principalement la notion de mémoire. En effet, le déroulement futur de l'algorithme est régi par un ensemble de règles déduites des informations de son passé. Ce chapitre présente les principes fondamentaux de la méthode. Dans la première section, nous définissons les principes de base qui caractérisent une méthode de recherche tabou. Cette version primaire présente toutefois des limitations. Dans la deuxième section, nous discutons des concepts additionnels qui, perfectionnent les méthodes de recherche tabou et leur confèrent plus de robustesse. Nous terminons en présentant les raffinements qui peuvent être apportés aux éléments de base dans le but d'accroître la performance de ces méthodes. Nous adopterons une structure qui se base sur l'article dû à Soriano et Gendreau [64]. Une étude plus détaillée ainsi que de nombreuses références sur le sujet s'y trouvent.

1.1 Principes de base: la mémoire à court terme

De manière analogue à Glover dans [25], nous partons du problème défini comme suit: nous désirons minimiser une fonction f sur un espace X de solutions. Notons x, une solution élément de X. Le voisinage $N(x)$ de x est défini comme l'ensemble des éléments de X qui peuvent être atteints en effectuant une transformation sur x. Dans les problèmes d'ordonnancement (voir [62]), ces transformations sont en général des permutations; dans les problèmes modelisés sur des graphes, ils correspondent souvent à des échanges de types Or-opt ou k-opt (voir [3] et [67]). On notera la meilleure solution jamais trouvée par l'algorithme x^* et la valeur correspondante de l'objectif, f^*. La méthode de recherche tabou peut être perçue comme une extension des méthodes de descente classiques. En effet, la recherche explore X en passant successivement d'une solution x à une solution $x' \in N(x)$. Dans une méthode de descente, x' deviendra la solution courante seulement si la valeur correspondante de l'objectif, $f(x')$, est meilleure que $f(x)$ (où dans ces méthodes, la solution courante est toujours la meilleure jamais trouvée, c'est-à-dire $f(x) = f^*$ et $x = x^*$). S'il est impossible de trouver une telle solution, l'algorithme s'arrête. La solution finale est un *optimum local*, soit x^* dans ce contexte. Dans une approche tabou, x' doit seulement correspondre à la plus petite valeur de f dans $N(x)$, même si $f(x')$ est moins bonne que $f(x)$. Cette politique évite d'être bloqué dans un optimum local, puisque le critère de choix d'une nouvelle solution ne dépend plus de $f(x)$. De plus, l'algorithme a dorénavant plus de chances de trouver l'optimum global puisqu'un plus grand nombre de solutions (dont plusieurs optima locaux) seront examinées. Cette modification pose néanmoins un nouveau problème. Dans une descente classique, l'algorithme évolue tant qu'il est possible de réduire la valeur de f^*. Il est donc certain que les solutions visitées seront toujours différentes. Dans la méthode de recherche tabou par contre, la seule condition pour visiter une solution est qu'elle soit *localement* la meilleure. Il devient alors possible de visiter les mêmes solutions plusieurs fois et de finir par cycler. La résolution de cette difficulté sera la motivation de l'utilisation

du concept de mémoire à court terme. En effet, supposons que nous partons d'une solution x à un stade donné dans l'algorithme. On effectue une transformation m sur x qui résulte dans $x' = m(x)$. Dans le futur, si l'algorithme se *rappelle* que m a déjà été effectué, cela permettra de ne pas appliquer la transformation inverse (m^{-1}) qui nous ramènerait à x. Ainsi, chaque fois qu'une modification est effectuée, son inverse est *mémorisé* et déclaré tabou ou interdit. Le risque de cycler sera donc réduit. Cependant, les transformations tabous ne le seront pas de manière permanente. En effet, si tel était le cas, il arriverait un moment où toutes les transformations (ou la plupart) seraient tabous, et la recherche stagnerait. C'est à cause de cette durée limitée des statuts tabous qu'on parle de mémoire à court terme.
Maintenant que l'idée de base de la recherche tabou a été définie, nous utiliserons le problème d'ordonnancement des tâches (étudié entre autres dans Song [62] et Woodruff [71]) pour en illustrer le fonctionnement.

1.1.1 Exemple: Le problème d'ordonnancement des tâches

Nous adopterons une notation semblable à celle de l'exemple dans Glover [27]. Supposons qu'on a n tâches j_i, $i = 1, 2, \ldots, n$. Pour chaque i donné, j_i s'exécute en un temps t_i et doit être complétée avant une date d_i. Si ce délai n'est pas respecté, une pénalité de retard c_i sera encourue. Notons e_i la date où la tâche j_i est effectivement achevée. Le but du problème est de trouver un ordonnancement des tâches qui minimise la pénalité totale de retard. De plus, il existe des contraintes de précédence définies dans un ensemble P de couples. Si le couple (j_i, j_k), $i, k = 1, 2, \ldots, n$ appartient à P, alors la tâche j_i doit être effectuée avant la tache j_k, de manière analogue à la définition de Maffioli et Sciomachen dans [45]. L'espace des solutions du problème est formé de tous les ordonnancements possibles des n tâches. Le voisinage d'une solution est défini en interchangeant la position de deux tâches. Notons $m(i,j)$ la modification qui consiste à positionner la tâche i à la place de la tâche j et vice-versa. Supposons qu'on a une solution $x = (1, 3, 5, 2, 6, 8, \ldots, n)$. Si $m(3, 6)$ est appliqué à x,

on obtient $x' = (1, 6, 5, 2, 3, 8, \ldots, n)$. Dorénavant, il est interdit, pendant un certain temps de positionner la tâche 6 à la place actuelle de la tâche 3, et vice-versa. *m(6,3)* sera déclaré tabou pendant un certain nombre d'itérations. Les structures utilisées pour réaliser ce mécanisme sont des listes tabous. Dans le paragraphe suivant, nous présentons les différentes formes de listes tabous ainsi que l'impact correspondant dans l'algorithme.

1.1.2 Les listes tabous

Ces structures sont des mémoires permettant de déclencher et d'annuler les statuts tabous dans l'algorithme. Il existe plusieurs alternatives quant à leur conception.
L'approche la plus naturelle consiste à mémoriser carrément les solutions déjà visitées. Cette approche garantit en tout temps de ne pas cycler puisque de telles solutions seront tabou pour toute la durée de la recherche. Dans l'exemple précédent, même si la solution x est à nouveau générée, elle sera ignorée et son voisinage ne sera plus exploré. Cette méthode sera néanmoins lourde à implanter et surtout, coûteuse en espace mémoire puisque les solutions visitées devront être entièrement mémorisées (pour pouvoir être interdites).
On utilise plutôt des listes circulaires pour représenter les interdictions. Comme leur nom l'indique, elles permettent de maintenir les statuts tabous de manière cyclique. Supposons que la liste tabou est notée T et contient $|T|$ éléments de mémorisation. Cette approche garantit que l'algorithme ne cyclera pas pendant au moins $|T|$ itérations à partir de la solution courante. De fait, à chaque itération, le statut tabou du dernier élément de T sera désactivé. La place ainsi libérée dans la liste permettra d'activer le statut tabou correspondant à la nouvelle solution courante.
Pour les raisons évoquées au paragraphe précédent, plutôt que de mémoriser des solutions complètes, il est préférable de définir ces tabous sur les transformations effectuées ou sur les attributs des solutions déjà visitées.
Dans le problème d'ordonnancement des tâches par exemple, les restrictions sont

définies sur les transformations. Après avoir adopté comme nouvelle solution courante x', la transformation qui ramènerait à x devient tabou.

Comme souligné dans [27], [26] et [28], la longueur des listes est un paramètre important pour la qualité des résultats de l'al-gorithme tabou. Si cette valeur est trop grande, l'algorithme pourrait stagner ou rater des solutions intéressantes, puisque les transformations resteraient interdites trop longtemps. Si elle est trop petite, l'algorithme cyclera rapidement. La longueur des listes est fréquemment déduite à partir de tests empiriques. Elle sera en général dépendante de l'ordre de grandeur du problème ou du voisinage. Elle peut être fixée pour toute la durée de l'algorithme comme dans [23].

L'élément sur lequel les tabous seront définis est très important pour leur efficacité. Dans l'exemple précédent, empêcher que les positions des tâches 3 et 6 soient interchangées ne garantit pas de ne plus retourner à x. En effet, supposons qu'à partir de x' l'algorithme opère la séquence de transformations m(6,5), m(2,3), m(6,2), m(5,8), m(3,8), m(2,5) et m(8,2). On obtient respectivement les solutions suivantes:
(1, 5, 6, 2, 3, 8, ... , n), (1, 5, 6, 3, 2, 8, ... , n), (1, 5, 2, 3, 6, 8, ... , n),
(1, 8, 2, 3, 6, 5, ... , n), (1, 3, 2, 8, 6, 5, ... , n), (1, 3, 5, 8, 6, 2, ... , n),
(1, 3, 5, 2, 6, 8 ... , n). On retombe donc sur x sans violer de statut tabou. Cet exemple montre que pour assurer le bon fonctionnement des restrictions, elles doivent être définies de manière rigoureuse.

Ainsi, après l'obtention de x', on pourrait interdire l'emplacement de la tâche 3 à la position 2 et celui de la tâche 6 à la position 5. Dans des problèmes plus compliqués, de telles interdictions nécessitent souvent l'utilisation de plusieurs listes tabous (Soriano et Gendreau [63], Séguin [60]). Si le cyclage est désormais évité, des combinaisons comme (1, 3, 2, 5, 6, 8, ... , n) ou (1, 3, 8, 5, 6, 2, ... , n), qui n'ont jamais été visitées, seront aussi exclues. Les statuts tabous peuvent donc également éliminer la candidature de solutions qui n'ont pas encore été visitées. Malheureusement, ces dernières pourraient être de bonne qualité ou au moins avoir un voisinage promet-

teur. Il faut donc trouver une mesure qui permettra de préserver des statuts tabous efficaces sans pour autant éliminer des solutions prometteuses, non explorées. Le critère d'aspiration a été introduit parmi les principes de la méthode de recherche tabou pour résoudre cette question.

1.1.3 Le critère d'aspiration

Le critère d'aspiration est une règle qui permet de révoquer un statut tabou. Cette décision dépend des conséquences qu'elle aura sur l'avenir de la recherche. Sous une forme simple, elle autorise l'examen d'une solution tabou x lorsque $f(x) < f^*$ dans un contexte de minimisation. Malheureusement, lorsque f^* correspond déjà à une très bonne valeur, cette règle sera rarement vérifiée et le problème demeurera. Dans une approche plus élaborée, on définit des niveaux d'aspiration. Supposons qu'on désire passer de x à x' qui est tabou. Puisqu'on ignore à priori si x' a déjà été visitée, on définit une borne pour $f(x')$ qui garantira que l'algorithme ne l'a pas encore examinée. Cette borne notée *A(f(x))* est le *niveau d'aspiration* de x' et si $f(x') \leq A(f(x))$, son statut tabou est révoqué sans risque de cycler. En général *A(f(x))* est une valeur plus petite que $f(x)$. Dans les problèmes à variables entières, on adopte $A(f(x)) = f(x) - 1$ (voir [31]). x' doit donc être *localement* la meilleure solution pour être acceptée. Des applications de ce concept se trouvent dans [60], Lokketagen [44], Lee et Koh [41]. L'exploration locale du voisinage régie par une mémoire à court terme de tabous constitue une recherche tabou dans sa forme fondamentale, les interdictions étant altérées par la règle établissant le critère d'aspiration. Le paragraphe suivant présente une forme générale de l'algorithme tabou où ces idées seront regroupées.

1.1.4 L'algorithme tabou

Nous adopterons la notation définie au tout début de cette section.

Algorithme

Initialisation

Choix d'une solution initiale x_0

Initialisation de la solution courante $x_c = x_0$

Initialisation de la meilleure solution $x^* = x_c$ **et** $f^* = f(x_c)$

Itération de l'algorithme

Tant que le critère d'arrêt n'est pas vérifié:

Appliquer le critère d'aspiration pour lever les tabous appropriés dans $N(x_c)$

Choisir $\bar{x}$ dans $N(x_c)$ telle que $f(\bar{x}) \leq f(x)\ \forall\ x\ \epsilon\ N(x_c)$ et x non tabou

Si $f(\bar{x}) < f^*$, poser $x^* = \bar{x}$ et $f^* = f(x_c)$

Activer le statut tabou de x_c et poser $x_c = \bar{x}$

Idéalement, on voudrait que la procédure s'arrête après l'identification d'un optimum global. Malheureusement, de telles solutions optimales sont inconnues pour la plupart des problèmes. En général, la condition d'arrêt pour l'algorithme sera une borne supérieure sur le temps total d'exécution, le nombre total d'itérations ou le nombre d'itérations sans améliorer f^*.

1.2 Éléments additionnels: la mémoire à long terme

L'utilisation d'une mémoire à court terme (liste tabou) réduit seulement le risque de cycler dans un algorithme tabou. C'est donc un outil qui ne garantit pas toujours la qualité des résultats. Il est possible d'accroître la performance de cette méthode en utilisant des informations plus détaillées sur son passé pour diriger son déroulement futur. La fiabilité de telles données nécessite cependant une observation de l'algorithme sur une période de temps d'exécution plus significative. Ainsi, la notion de mémoire est encore mise à contribution mais sur un long terme cette fois-ci. Dans cette section,

nous verrons comment et dans quels contextes une recherche tabou pourrait exploiter ce principe.

1.2.1 La diversification et l'intensification

Dans la section 1, nous avons vu que la définition du voisinage et la longueur des listes tabous sont des éléments déterminants pour la performance de la RT. Bien souvent, l'algorithme tabou n'améliore pas la meilleure solution trouvée pendant une longue période de temps. Cela ne signifie pas toujours qu'on est sur un optimum global. En effet, il peut exister de meilleures solutions qui demeurent inaccessibles à cause des tabous par exemple (longueur de la liste). Pour régler ce problème, le concept d'intensification (voir [26]) a été introduit dans la recherche tabou. L'idée consiste à approfondir la recherche dans une région de l'espace jugée prometteuse. D'un autre côté, la trajectoire empruntée jusque-là par l'algorithme pourrait être très localisée. La région correspondante de l'espace des solutions est alors presque totalement explorée. Afin de poursuivre la recherche de bonnes solutions dans d'autres régions de l'espace des solutions, on aura recours à une procédure de diversification (voir [26]). Cette dernière a pour objectif de rediriger la recherche vers des parties de l'espace des solutions qu'on estime trop peu explorées ou inexplorées.

À cause de leur but bien déterminé, l'application de ces deux types de procédures nécessite une bonne compréhension de l'espace de recherche, ainsi qu'une caractérisation approfondie des solutions prometteuses et des régions explorées jusque là. Dans la diversification par exemple, la manière la plus commune de procéder est en relançant l'algorithme à partir d'une nouvelle solution. Cette dernière devra donc induire un voisinage qui n'a jamais été visité. Mais comment déterminer une telle solution? Il existe plusieurs possibilités dont la plupart exploitent les informations du passé de la recherche. Dans [63], on tente de résoudre le problème de la clique maximum. Pour effectuer une diversification, un sommet qui n'a pas encore été visité est utilisé afin de former une nouvelle solution de départ. Dans Hubsher et Glover

[31], la diversification est appliquée de manière déterministe en effectuant des transformations *influentes* sur les solutions de manière à les diversifier. L'algorithme est relancé à partir des solutions ainsi obtenues. Dans Chakrapani et Skorin-Kapov [61], la recherche est relancée en construisant une nouvelle solution à partir des attributs les moins souvent utilisés; de plus, les attributs fréquemment utilisés seront sciemment classés tabous pour une période de temps assez longue pour permettre l'exploration des nouvelles régions.

Lorsqu'on désire éviter un changement trop brutal dans l'exploration des solutions, il existe des façons plus subtiles de procéder. La diversification continue consiste à diriger *implicitement* la recherche vers de nouvelles régions sans avoir à donner un nouveau point de départ à l'algorithme. Dans [63] par exemple, on mémorise le nombre d'itérations consécutives depuis qu'un attribut est présent dans chaque solution visitée. Le nombre total d'itérations pendant lequel un attribut a intégré les solutions visitées est également conservé. Dans le premier cas, on parle de mémoire de résidence (ou mémoire temporelle) et dans le second, il s'agit d'une mémoire de fréquence. Pour éliminer les attributs trop souvent présents dans les solutions, un terme de pénalité proportionnel à la mémoire temporelle est ajouté dans la fonction objectif. De même, pour exclure les attributs trop fréquents dans les solutions déjà visitées, un terme de pénalité proportionel à leur fréquence sera intégré dans la fonction objectif. Ainsi, les fréquences serviront égalemment à encourager les attributs rarement utilisés. Dans Woodruff et Spearman [72], un terme de pénalité est utilisé de manière semblable.

Dans l'intensification, des manoeuvres similaires se basant sur la mémoire sont pratiquées. La plus simple consiste à recommencer l'algorithme à partir de la meilleure solution jamais générée. C'est le cas dans [60] où on intensifie la recherche autour de la meilleure solution en augmentant la taille du voisinage. Une approche similaire est appliquée dans Lee et Koh [41]. Il existe des approches plus poussées. L'intensification par des *solutions élites* consiste à trier les solutions de bonne qualité parmi celles visitées et à repartir l'algorithme à partir de ces dernières. Le meilleur résulat sera

retenu. Cette approche a été appliquée avec succès dans Laguna et al. [35] et dans Armentano et Roconi [2]. L'intensification par décomposition est une autre possibilité définie dans Glover [27]. Ici, lorsque les solutions élites sont déterminées, des restrictions sont imposées sur certaines de leurs composantes, afin de concentrer la recherche sur les autres parties jugées plus prometteuses.

1.2.2 L'oscillation stratégique

Il arrive que l'algorithme tabou se retrouve dans un creux d'optimum local et ait de la difficulté à s'en extirper. De plus, la définition du voisinage est souvent telle que la trajectoire permettant de s'en sortir et surtout de retrouver un optimum local de meilleure qualité (ou optimum global) est très longue. Pour régler cette difficulté, l'oscillation stratégique a été intégrée dans la recherche tabou. C'est une technique qui consiste à accepter dans l'exploration une détérioration notoire de la fonction objectif ou encore des solutions non réalisables. Cette tolérance est généralement contrôlée par un terme de pénalité dans la fonction objectif. Comme il a été démontré dans Rolland et al. [58], cette stratégie favorise un chemin plus rapide vers les vallées plus profondes de la fonction objectif. L'oscillation stratégique peut aussi être utilisée comme régulateur des stratégies de diversification et d'intensification. En effet, le terme de pénalité qui la matérialise dans la fonction objectif pourrait être calibré de manière à favoriser la recherche dans de nouvelles régions ou à l'intensifier dans une région précise. Dans [29] et [46], la diversification et l'intensification sont implantées en utilisant l'oscillation stratégique.

1.3 Raffinements

Dans cette section, nous présentons des versions plus élaborées des structures de base d'un algorithme tabou. Celles-ci apporteront généralement une amélioration dans la performance de la méthode.

1.3.1 Les listes tabous

La longueur des listes tabous est l'un des paramètres les plus importants de l'algorithme. Dans une étude comparative des métaheuristiques, Yagiura et Ibaraki [74] démontrent que l'efficacité de la méthode de recherche tabou dépend en grande partie de la longueur des listes tabous.

Il est bien plus simple de considérer cette valeur comme fixe, ce qui était fait habituellement dans les premières utilisations de la méthode. Cependant, plusieurs études ont montré qu'il est préférable que la longueur des listes soit variables (Taillard [65] et Friden et al. [17]).

On peut déterminer la longueur des tabous de manière dynamique. Ce paramètre sera alors changé après un nombre fixe d'itérations. Les différentes valeurs qu'il pourra prendre seront choisies de manière cyclique dans un intervalle $[T_{min}, T_{max}]$. Dans [31] par exemple, pour éviter de trop longues séquences de transformatios tabous, on utilise une liste dynamique. Pour obtenir le même résultat, on pourrait aussi générer des longueurs aléatoires. Dans [61], cette valeur est générée aléatoirement après qu'un certain nombre d'itérations a été atteint, sans améliorer f^*. Il existe des approches encore plus recherchées associant dynamisme et aléa. Par exemple le nombre d'itérations avant de changer la longueur de la liste tabou peut être aléatoire. Cette idée est développée de manière plus élaborée dans Glover [27].

1.3.2 L'exploration du voisinage

Lorsque la structure du problème traité est riche, le voisinage est vaste et son exploration aveugle peut s'avérer trop coûteuse. Plusieurs techniques ont été élaborées pour partitionner la recherche dans un voisinage. Leur but premier est de minimiser les risques d'exclusion des solutions de bonne qualité. En adoptant la notation définie à la section 1, supposons qu'à partir d'une solution x, l'algorithme explore $N'(x) \subset N(x)$ plutôt que $N(x)$. La manière la plus simple de définir $N'(x)$ est de le générer aléatoirement. Néanmoins, si cette approche est appliquée sans biais, on

risque alors de se retrouver avec beaucoup de mauvaises solutions.

Des techniques plus intelligentes sont généralement pratiquées. Les listes de candidats permettent de sélectionner un sous-ensemble de $N(x)$ et d'explorer uniquement le voisinage des solutions ainsi obtenues. Cette technique est utilisée dans [31] et [2]. Plusieurs techniques peuvent être mises en pratique pour déterminer une liste de candidat. Dans [51] et [52], on choisit des solutions prometteuses et l'exploration sera limitée à leur voisinage. Dans [59] et [34] on examine à chaque itération un sous-ensemble déterminé du voisinage. Les solutions inexplorées feront partie du voisinage aux itérations suivantes. Il s'agit donc d'une partition cyclique. Dans les travaux [24] et [27] dûs à Glover, il existe des développements plus exhaustifs sur les choix de conception des listes de candidats.

Chapitre 2

La recherche tabou appliquée aux problèmes stochastiques de tournées de véhicules

Dans ce chapitre, nous définissons les principes des problèmes de tournées de véhicules cités dans l'introduction, étant donné que la diversification présentée dans ce mémoire y est appliquée. Après avoir présenté chaque problème, nous exposerons quelques approches de résolution par la recherche tabou (RT), proposées dans la littérature.

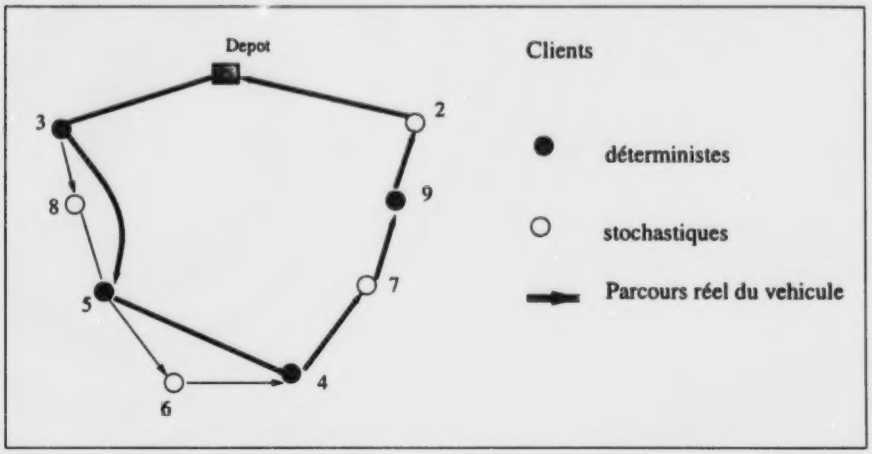
2.1 Définition des problèmes

2.1.1 Le problème du voyageur de commerce avec clients stochastiques (PVCCS)

Le PVCCS est un PVC où la probabilité de présence dans un sous-ensemble de clients est inférieure à 1 pour chacun d'eux: il n'est pas acquis qu'ils nécessitent une visite à chaque exécution de la tournée. Le problème est que d'une fois à l'autre, les clients visités ne seront plus les mêmes. On aura donc plusieurs instances de la tournée à optimiser. De ce fait, la distance ou le temps nécessaire pour compléter la tournée deviennent des variables aléatoires. Au lieu de réoptimiser le parcours à chaque fois, sachant quels clients seront effectivement présents, on déterminera plutôt un ordre de parcours *a priori* qui sera toujours suivi, quelle que soit la réalisation des variables

aléatoires désignant la présence. On suppose que la présence des clients est connue seulement au début de la tournée. Le parcours du tour a priori est toujours respecté, tout en sautant les clients absents. Un tel parcours est montré à la Figure 2.1. Ce

Figure 2.1: Instance d'un tour dans le PVCCS



sont les économies imprévisibles dues à ces sauts qui introduisent de l'aléa dans le coût de parcours. Ainsi, il faut déterminer un tour a priori tel que l'espérance de son coût est minimale. Plus précisément, pour chaque tour possible, on détermine le coût total comme si tous les clients étaient toujours présents; de ce coût, est retranchée *l'économie espérée* résultant de l'absence des clients stochastiques du problème.

Les articles de Jaillet [32], Bertsimas et Howell [8], Laporte, Louveaux et Mercure [37] font état du sujet. Dans la dernière référence, la présence d'un sous-ensemble de clients suit une loi de probabilité uniforme. Comme dans la plupart des problèmes de tournées de véhicules stochastiques, la programmation stochastique en deux étapes est utilisée pour l'évaluation des tournées.

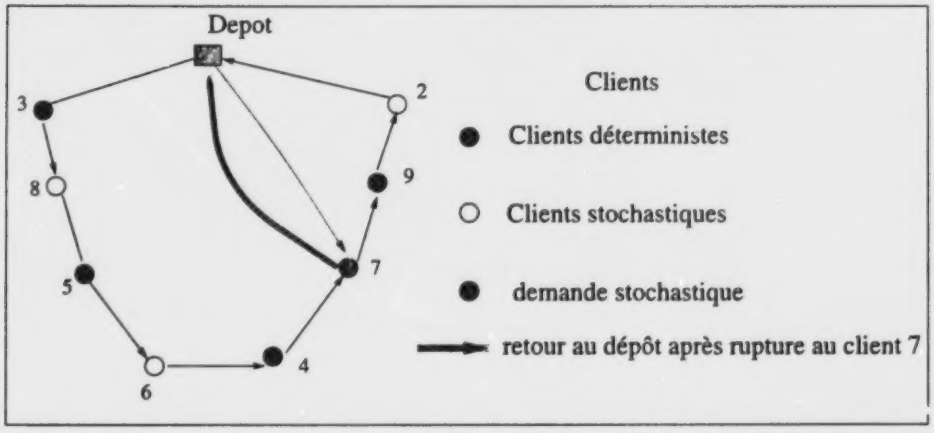
Par ailleurs, on peut trouver dans Bertsimas, Jaillet et Odoni [9] une description détaillée de l'optimisation a priori, ainsi qu'une comparaison avec la réoptimisation dans une analyse asymptotique. De plus, les articles de Bertsimas et Howell [8], Jaillet

[32], Laporte, Louveaux, Mercure [37] présentent une application de cette méthode au PVCCS.

2.1.2 Le problème de tournées de véhicules avec demandes stochastiques (PTVDS)

Dans un problème classique de tournées de véhicules, les demandes de chaque client sont connues à l'avance. Dans le PTVDS, elles suivent une loi de probabilité donc il faut optimiser chaque tournée sans être capable de prévoir l'évolution du chargement du véhicule. La composante du coût de la tournée qui devient une variable aléatoire est le *coût de recours*. Un coût de recours est encouru lorsque la capacité du véhicule est atteinte: celui-ci doit retourner au dépôt pour être rempli à nouveau et continuer sa tournée à partir de l'endroit où elle a été interrompue. On peut voir un exemple de rupture dans un parcours à la Figure 2.2. De plus, selon la politique de livraison

Figure 2.2: Rupture au client 7 dans le PTVDS



considérée, on peut définir plusieurs variantes du problème. En livraison partielle par exemple, lorsque la capacité résiduelle du véhicule est inférieure à la demande du prochain client qu'on doit servir, il est permis de la satisfaire partiellement pour la compléter après le ravitaillement au dépôt. Cette politique permet de ne pas

gaspiller la capacité résiduelle. En situation de livraison totale, les clients doivent toujours recevoir la totalité de leur demande ou rien du tout.

Étant donné que les demandes sont aléatoires, les points de rupture sont imprévisibles. Ainsi, comme dans le PVCCS, on désire déterminer un parcours a priori pour chaque tournée. Cette tournée doit être telle que la somme du coût de parcours pour visiter tous les clients et du coût espéré de recours associé est minimale.

Le PTVDS a été largement étudié depuis son introduction par Tilman [69]. Dans Bertsimas [8], on propose une solution du problème quand les demandes suivent une loi discrète. L'optimisation a priori est introduite en utilisant dans la fonction objectif un terme pour le coût de rupture. Deux stratégies de parcours sont suggérées: sauter ou ne pas sauter les clients dont la demande est nulle. Une heuristique permet de calculer efficacement le coût de recours. Dans Laporte, Louveaux et Van Hamme [38], le PTVDS est étudié quand les demandes suivent une loi Poisson et une loi normale. Dans ce dernier cas, étant donné que les demandes sont continues, la politique de livraison partielle est utilisée. On trouvera d'autres applications du PTVDS dans Bouzaiene-Ayari, Dror et Laporte [10], Dror, Laporte et Trudeau [15], Ong, Ang, Goh et Deng [49].

2.1.3 Le problème de tournées de véhicules avec clients et demandes stochastiques (PTVDCS)

Il s'agit d'un problème stochastique de tournées de véhicules où la présence et la demande de chaque client sont des variables aléatoires. La présence suit en général une loi de Bernouilli tandis que la demande peut être discrète (Gendreau, Laporte, Seguin [21]) ou continue (Laporte, Louveaux, Van Hamme [38]). Maintenant, deux éléments s'ajoutent à la logique du problème: on ignore quels clients seront présents et quand ils le sont, leur demande est également inconnue à l'avance. Donc d'une part, il y aura une économie due au saut des clients absents et d'autre part, un coût de recours dû à d'éventuels retours au dépôt lorsque la demande cumulative excède la

capacité du véhicule. Ces deux dernières composantes sont des variables aléatoires. Il faut donc déterminer pour chaque tournée un tour a priori tel que le coût total de parcours moins les économies espérées dues au saut des clients absents plus les coûts espérés de rupture de stock est minimal.

Tableau 2.1: Les problèmes stochastiques de tournées de véhicules.

Type	Composante Stoch.	Conséquence	Tour a priori
PVCCS	Clients	Saut des clients absents ⇒économies	Minimum: distance totale de parcours – économie espérée
PTVDS	Demandes	Rupture imprévue ⇒ coûts additionnels dûs aux retours au dépôt pour rétablissement de la capacité	Minimum: distance total de parcours + le coût espéré de rupture
PTVDCS	Clients et Demandes	Ruptures et sauts des clients absents ⇒ pénalités et économies	Minimum: distance totale parcourue – économie espérée + le coût espéré de rupture

2.2 Application de la recherche tabou aux problèmes stochastiques de tournées de véhicules

La recherche tabou (RT) a été largement utilisée pour la résolution des problèmes de transport comme nous le verrons à la prochaine section. Cependant, très peu d'attention a été consacrée aux problèmes de tournées de véhicules avec composantes stochastiques: présence des clients, demandes des clients, temps de service des clients ou différentes combinaisons des trois. L'une des rares références sur la RT appliquée aux PTV avec clients et demandes stochastiques se trouve dans Séguin [60] (voir aussi Gendreau, Laporte et Séguin [21]). Étant donné que la diversification présentée dans ce mémoire se base sur cet algorithme de RT, nous présenterons son application dans le contexte particulier du PTVDCS.

Du point de vue structurel, cet algorithme est très flexible: il admet des solutions non réalisables et la plupart des valeurs importantes pour l'efficacité (longueur des listes

tabous, nombre de routes, fonction de pénalité) sont paramétrisées ou exprimées en fonction de la taille du problème ou du temps (nombre d'itérations). Cela accroît les aspects dynamiques et intelligents de la recherche.

La solution initiale est constituée uniquement d'aller-retours vers chaque client, elle est donc très probablement non réalisable, puisque le nombre total de tournées sera en général supérieur à celui permis ou à la valeur maximale permise. Notons que dans notre contexte, le nombre de véhicules, bien que paramétrisé, n'est pas une variable, sujette à une borne supérieure. Il est considéré fixe pour tous les problème. Un terme de pénalité sera introduit dans la fonction objectif de base pour les solutions non réalisables. Pour une solution non-réalisable, la pénalité est proportionnelle au nombre de tournées additionnelles ou manquantes. Elle dépendra aussi d'un facteur autocorrecteur dynamique, dont la valeur s'ajuste en fonction du nombre de solutions non-réalisables générées dans un passé relativement proche.

Remarquons qu'on aurait pu utiliser une méthode de descente classique – algorithme vorace par exemple – pour générer une solution de départ. Cependant, étant donné que la RT elle même se base déjà sur une méthode dedescente locale, cette approche ce serait avérée inutile. Par ailleurs, partir d'une solution non réalisable augmente les chances d'explorer une plus grande portion de l'espace des solutions. En effet, une telle solution aura une très mauvaise valeur dans la fonction objectif, à cause des pénalités. L'amélioration de cette valeur nécessitera donc la visite d'un grand nombre de solutions, ce qui ira de pair avec la détermination d'une solution réalisable.

Le voisinage d'une solution x est généré en déplaçant successivement q sommets avant et après (la position correspondant à la plus petite valeur de la fonction objectif sera retenue) leur p voisins les plus proches. Le dépôt n'est jamais déplacé. La valeur de q dépend du nombre de clients et du nombre de routes tandis que la valeur de p dépend seulement du nombre de clients. Les mouvements qui permettent l'exploration du voisinage peuvent créer une nouvelle tournée, en détruire une déjà existante (il s'agit d'un aller-retour dans les deux cas) ou encore, conserver la même structure de

solution. Dans ce cas, un sommet est déplacé dans sa propre tournée ou vers une tournée déjà existante.

La mémoire à court terme est appliquée en utilisant des listes tabous circulaires. Les listes tabous conservent les attributs des mouvements plutôt que des solutions entières. Comme souligné au premier chapitre, plusieurs listes tabous peuvent être utilisées pour définir des restrictions plus élaborées ou plus sévères sur un mouvement. C'est le cas ici lorsqu'un sommet est déplacé dans une route déjà existante. À titre d'exemple, supposons que le sommet x_i est déplacé dans une tournée r_y – qui peut être sa tournée actuelle ou une autre tournée – avant ou après un sommet y_i; la première liste tabou permet l'interdiction de tout déplacement de x_i à partir de r_y et la deuxième liste tabou permet l'interdiction du déplacement de y_i dans r_y avant ou après x_i *seulement* si les deux sommets se succédaient avant le mouvement. La longueur l des listes tabous est une variable aléatoire uniformément générée dans un intervalle dont la borne supérieure dépend du nombre total de clients. Un simple critère d'aspiration révoque le statut tabou d'un mouvement qui résulterait en une solution dont la valeur correspondante de la fonction objectif est inférieure à celle de la meilleure solution générée jusque-là.

Un élément de mémoire à long terme de la RT intervient dans la dernière étape de l'algorithme. Une intensification est executée lorsque l'algorithme tabou initial excède le nombre maximal d'itérations permises sans améliorer la meilleure solution. La variante choisie à cet effet recommence l'algorithme à partir d'une seule solution élite: la meilleure solution jamais générée. Une nouvelle trajectoire de recherche est ainsi créée. Les statuts tabous sont effacés mais en contrepartie, le voisinage est agrandi: la fonction qui permet de déterminer la valeur du paramètre de voisinage p produit des plus grandes valeurs. De plus, durant cette phase, tous les sommets sont déplacés auprès de leurs plus proches voisins, ce qui permet d'exécuter plus de mouvements prometteurs. La stratégie d'intensification appliquée ici s'apparente donc aux deuxième et troisième variantes décrites au Chapitre 1.

L'autre aspect de l'utilisation de mémoire à long terme appliquée aux méthodes de RT est la diversification. Le chapitre suivant décrit la diversification que nous avons ajoutée à cet exemple d'algorithme tabou.

2.3 Application aux autres problèmes de tournées de véhicules

La recherche tabou est généralement très utilisée dans la résolution d'autres types de problèmes de transport. La mémoire à court terme est toujours utilisée sous forme de liste tabou. Cette dernière contient les attributs des mouvements effectués ainsi qu'un nombre d'itérations pendant lequel ils ne doivent pas être utilisés. Presque toujours, la longueur du statut tabou est tirée aléatoirement dans un intervalle donné. La mémoire à long terme est aussi mise à contribution et dans beaucoup de cas, à des fins de diversification. En effet, en se basant sur la fréquence des mouvements, un terme de pénalité est introduit dans la fonction objectif pour décourager ceux qui sont trop souvent effectués. Ces politiques ont été expérimentées dans Gendreau, Hertz et Laporte [18]. Dans Rochat et Taillard [57], on met en oeuvre des stratégies de diversification et d'intensification à partir des solutions de l'algorithme tabou de base. Dans Xu et Kelly [73], la liste tabou impose des restrictions plus sévères. Les mouvements du voisinage sont des insertions et des interchanges de sommets dans les routes. En plus de l'interdiction d'être à nouveau mutés dans la route d'où ils viennent, les sommets ne doivent plus être déplacés du tout s'il l'ont été récemment. On peut trouver des exemples d'application de la RT plus récents dans Gendreau, Laporte et Musaraganyi [19], Crainic, Gendreau et Farvolden [12], Hertz, Laporte et Mittaz [30] et Gendreau, Laporte et Potvin [20].

Chapitre 3

La procédure de diversification

Tel que mentionné plus tôt, nous mettons en oeuvre une approche de diversification dans ce mémoire, le but étant d'améliorer la performance d'une recherche tabou simple. Pour ce faire, la meilleure solution réalisable trouvée par l'algorithme de base sera modifiée par un opérateur de voisinage plus agressif: le *cross-exchange*. Ensuite, l'algorithme sera relancé à partir de la solution résultante. Étant donné qu'il s'agit de diversifier, la mémoire à court terme (statuts tabous) sera mise à jour en fonction des changements effectués par la diversification.

3.1 Approche générale: le cross-exchange

Le cross-exchange est une approche de génération de voisinage bien adaptée aux problèmes de tournées de véhicules. Le principe de base est la modification de l'ordre de parcours des sommets. Cette méthode a été entre autres utilisée dans [66] pour la résolution d'un problème de tournées de véhicules avec fenêtres de temps.
Soit x, une solution d'un problème de tournées de véhicules. Soit deux séquences de sommets disjointes S_1 et S_2 appartenant à une ou des tournées dans x. Le cross-exchange consiste à interchanger la position des séquences S_1 et S_2. Précisément, on remplace S_1 par S_2 dans la tournée de S_1 et S_2 est remplacée par S_1 dans sa tournée. Lorsqu'elles appartiennent à une même tournée (voir Figure 3.1), on dénommera

Figure 3.1: Cross-exchange à l'intérieur d'une route: Intra-route

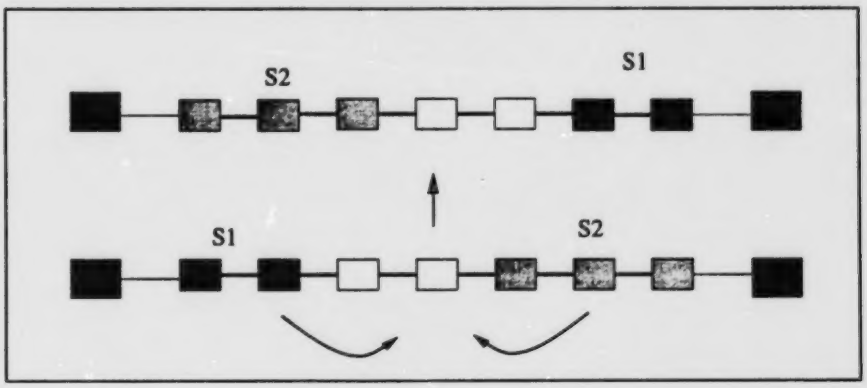
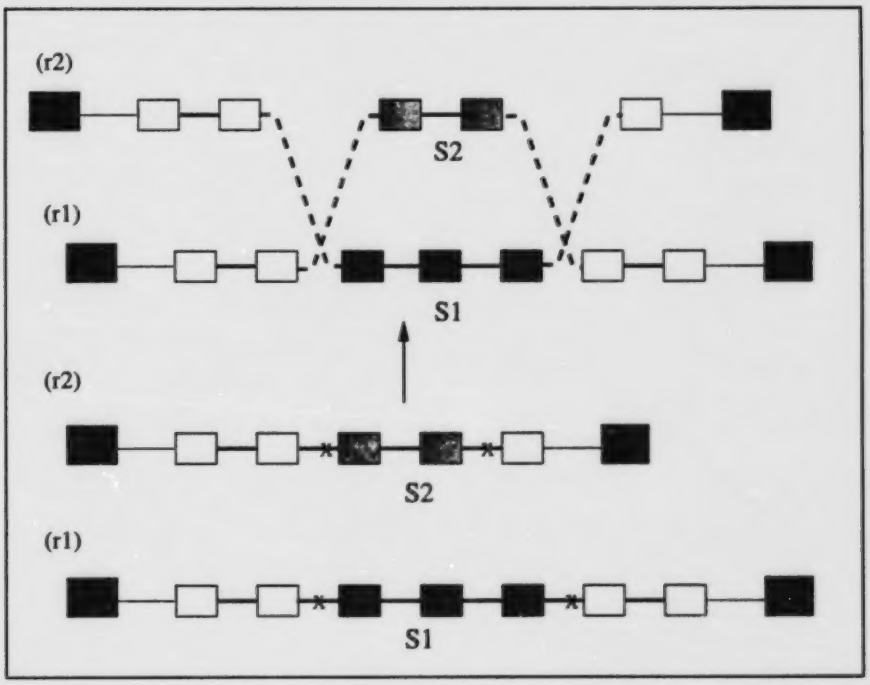


Figure 3.2: Cross-exchange entre 2 routes distinctes: Inter-routes



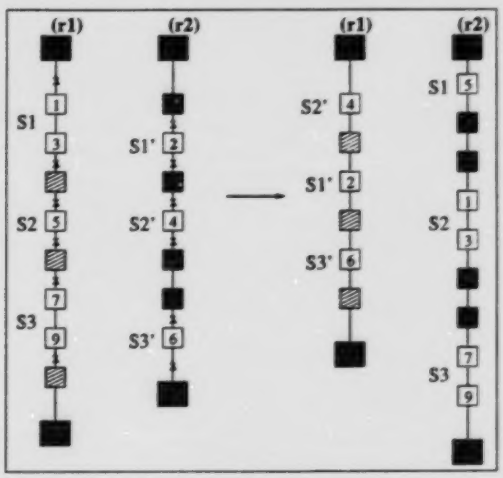
l'opération *intra-route*. Si S_1 et S_2 proviennent de 2 tournées distinctes, le cross-exchange sera appelé *inter-routes* (voir Figure 3.2). Comme le montrent les Figures 3.1 et 3.2, le sommet précédent le premier sommet de S_1 sera dorénavant suivi du premier sommet de S_2 et vice-versa pour S_2. De même, le sommet suivant le dernier sommet de S_1 sera maintenant précédé du dernier sommet de S_2.

3.1.1 Variantes du cross-exchange

Généralisation

Lorsque le nombre de sommets dans les tournées est relativement grand, le cross-exchange peut être généralisé à un échange entre plus d'une paire de séquences de sommets. À la Figure 3.3, on voit 3 paires de séquences où les séquences échangent leur place respective. Dans certains cas, cette approche est utile si l'on désire une diversi-

Figure 3.3: Cas général du cross-exchange: plusieurs paires de segments sont impliquées



fication plus agressive. Son application nécessite toutefois une bonne connaissance du comportement du problème. Dans le contexte stochastique surtout, l'application d'un tel opérateur pourrait complètement détruire la structure de la solution courante. En

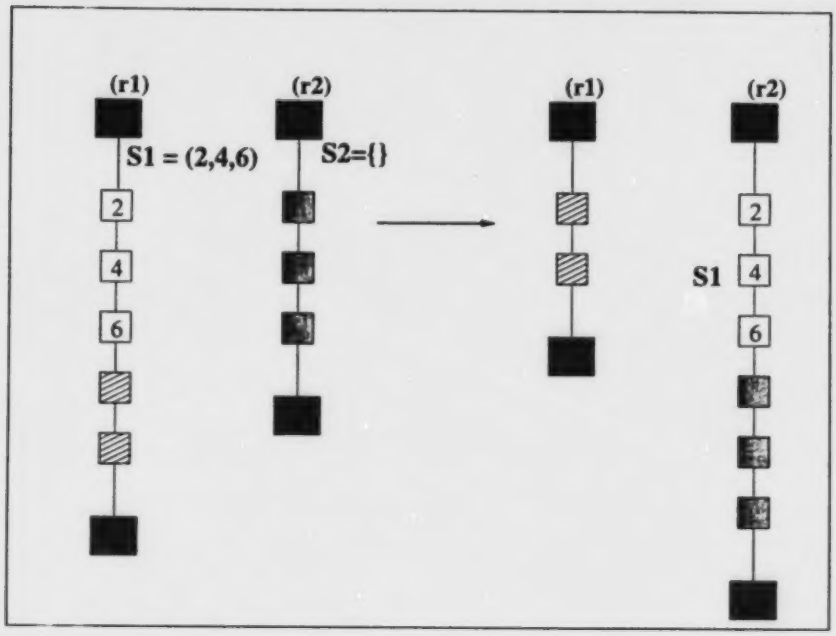
effet, quand les demandes ou les clients sont stochastiques, l'ordre de parcours est très important dans la caractérisation des solutions. Donc si on désire diversifier tout en conservant certaines structures de la solution de départ certaines précautions seront nécessaires. L'approche la plus simple consiste à mettre le moins de sommets possible dans les séquences impliquées dans chaque cross-exchange. L'effet demeurera diversificateur puisque maintenant, il y a plusieurs paires de séquences. De façon plus élaborée, on pourrait s'arranger pour déplacer les sommets les "moins stochastiques", c'est-à-dire, ceux qui ont une forte probabilité de présence ou une faible demande. Par ailleurs, quand la diversification est appliquée à partir d'une solution de bonne qualité par exemple, on désire souvent conserver quelques attributs prometteurs (voir chapitre 1). En conclusion, une simple diversification mécanique avec cet opérateur généralisé serait risquée dans le cas stochastique. Il serait plus prudent d'y annexer un mécanisme examinant les sommets impliqués dans le cross-exchange.

Les transferts cycliques dans [68], sont une autre forme de généralisation du cross-exchange. Supposons qu'on désire effectuer une rotation $\Phi = (3, 1, 2)$. Cela signifie qu'une séquence de sommets S_3 extraite de la tournée 3 doit être transférée à la tournée 1. Ensuite, une séquence S_1 de la tournée 1 différente de S_3 sera transférée vers la tournée 2. La même opération sera répétée de la tournée 2 vers la tournée 3 pour boucler la rotation. Le cross-exchange est un transfert cyclique dans lequel le nombre de tournées dans Φ est 2.

Quelques cas particuliers

On peut aussi envisager une variante du cross-exchange où une des séquences de la paire ne contient aucun sommet. Une telle séquence sera dite de cardinalité nulle. Par exemple, supposons que $|S_1| = 0$ et $|S_2| \leq 3$, le mouvement obtenu est un échange de type Or-opt ([50]) comme on le voit dans Figure 3.4.

Figure 3.4: Cas particulier du cross-exchange: une seule séquence change de route

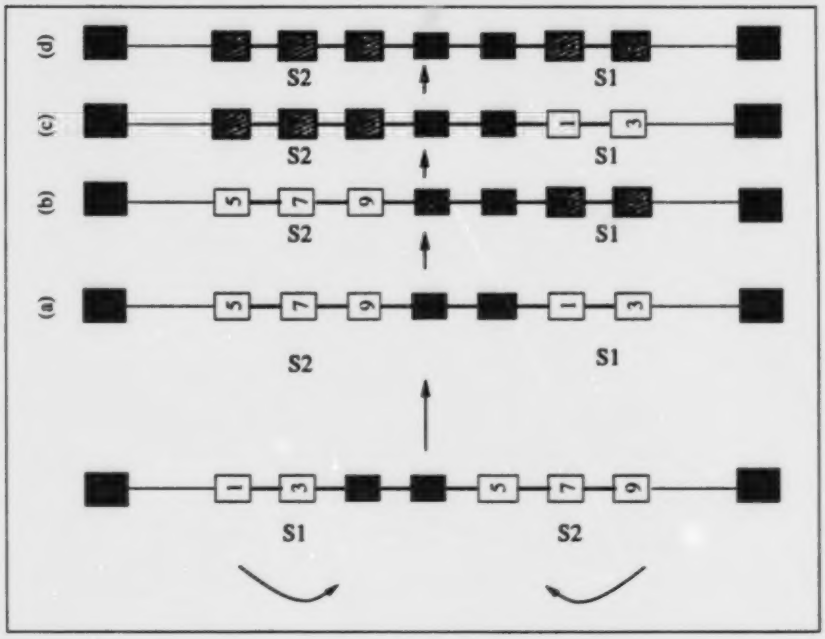


3.1.2 Mécanisme et précautions

L'intra-route

Dans cette application du cross-exchange, une seule tournée est impliquée. Afin de retenir le meilleur résultat possible, 4 scénarios d'ordre de parcours de S_1 et S_2 sont envisagés. La première alternative correspond aux sens de parcours de S_1 et S_2 tels quels dans la solution de départ (Figure 3.5a). À la Figure 3.5b, seul le

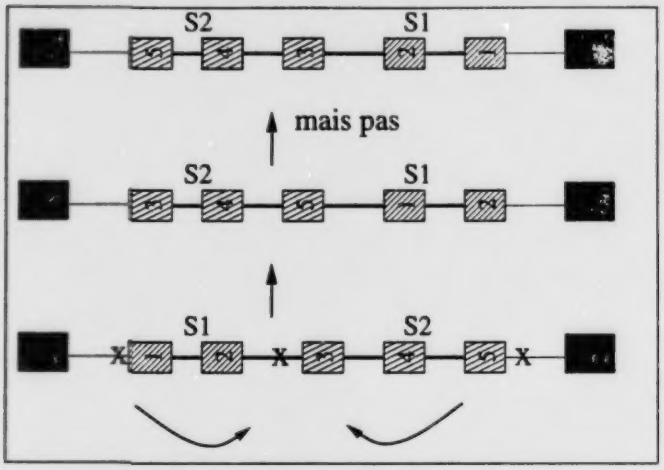
Figure 3.5: Sens de parcours des segments interchangés dans l'intra-route



sens de parcours initial de S_1 est inversé. La même opération est appliquée à S_2 seulement, dans la Figure 3.5c. Enfin, les sens de parcours initiaux des 2 séquences sont simultanément inversés dans Figure 3.5d.

Dans l'intra-route, il est primordial que le cross-exchange s'effectue sur des séquences contenant au moins 2 sommets chacune. En effet, s'il est permis à un seul sommet d'être déplacé, on obtient un simple mouvement de l'algorithme tabou de base. Par

Figure 3.6: Sens de parcours à ignorer dans l'intra-route

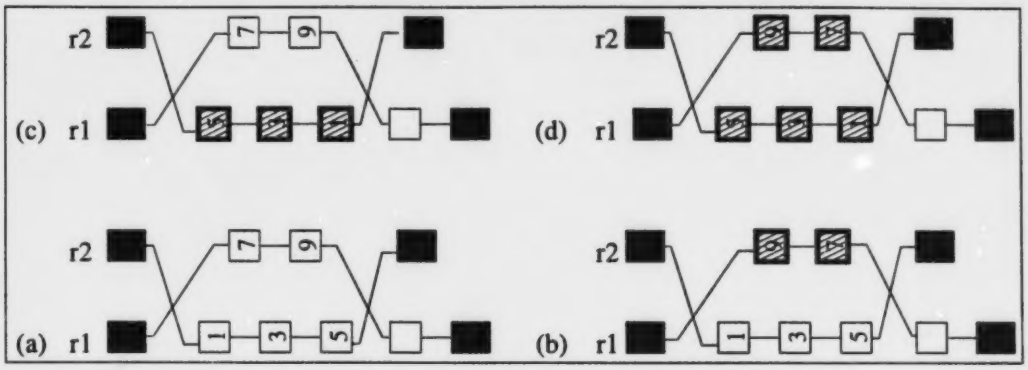


ailleurs, si S_1 et S_2 ensembles regroupent *tous* les sommets de leur route, l'alternative de la Figure 3.5d ne doit pas être considérée à l'évaluation du résultat. En effet, on retrouverait la solution initiale, (voir Figure 3.6) puisque comme mentionné au chapitre 2, l'algorithme tabou de base évalue chaque tournée en considérant ses 2 sens de parcours possibles.

L'inter-routes

Dans ce cas, les 4 scénarios décrits à la section précédente sont également envisagés. À la figure 3.7, les séquences dont les encadrements des sommets sont gras ont leur sens de parcours initial inversé. Ainsi, dans le cas a, les sens de parcours sont inchangés. Le cas le plus économique sera retenu. Au moins une des 2 séquences doit contenir plus d'un sommet, sinon, on effectue un simple mouvement de l'algorithme tabou de base. De plus, il faut imposer d'autres restrictions pour fins de cohérence. Supposons que le nombre total de sommets dans la tournée contenant S_1 est t_1 (dépôt exclus) et que de manière analogue pour S_2, ce soit t_2. On ne doit jamais avoir $|S_1| = t_1$ et $|S_2| = t_2$ (ou $|S_1| = 0$ et $|S_2| = 0$) en même temps. En effet, on se retrouverait avec

Figure 3.7: Sens de parcours dans l'inter-routes



la solution de départ.

Par ailleurs, supposons qu'on a le cas où l'une des séquences ne contient aucun sommet, disons $|S_1| = 0$. Pour éviter d'effectuer un simple mouvement, on doit imposer la condition $|S_2| > 1$ et on ne doit jamais avoir $|S_2| = t_2$. En effet, cela signifie que tous les sommets de la tournée contenant S_2 sont transférés dans la tournée contenant S_1 (qui elle ne transfère aucun sommet). Dépendamment des hypothèses assumées dans la diversification, des difficultés ou des violations pourraient se présenter. Dans notre cas par exemple, une des conditions de validité des solutions est le nombre de tournées qu'elle contient. Ce nombre est fixe et prédéterminé. La diversification qui prend en entrée une solution réalisable ne doit en aucun cas retourner une solution non réalisable. Donc cette procédure ne peut détruire ou créer une autre tournée en effectuant des mouvements. Notons toutefois que puisque l'algorithme tabou de base que nous utilisons traite les solutions non réalisables (voir chapitre 2), cette contrainte aurait pu être omise.

3.2 Structures de données

Dans cette section, nous définissons uniquement les paramètres important dans l'exécution du cross-exchange. Il s'agit des sommets, des tournées et des listes tabous. Les autres données sont clairement décrites dans [60] et ne sont pas vraiment critiques dans la procédure de diversification en tant que telle. Un sommet i est caractérisé par:

- la route à laquelle il appartient. Le tableau $itourn(i)$ retourne le numéro de la route où il se trouve. Dans une solution contenant m routes, elles seront étiquetées de 1 à m;
- son successeur et son prédécesseur, donnés par les tableaux $succ(i)$ et $pred(i)$ respectivement;
- sa position dans la route où il se trouve. Le tableau $posit(i)$ retourne cette valeur.

Les structures *successeur* et *prédécesseur* permettront de se déplacer dans la tournée. La structure *itourn* sera utile lorsqu'après un inter-routes, il sera nécessaire de spécifier les nouvelles tournées des sommets impliqués. La structure *posit* permettra de localiser directement un sommet dans sa tournée, au lieu de la parcourir jusqu'à tomber sur ce sommet.

Une tournée l est caractérisée par le sommet qui précède le dépôt dans l (dernier sommet de l), et le sommet qui succède au dépôt dans l (premier sommet de l). Les tableaux $adj1(l)$ et $adj2(l)$ retournent les numéros des sommets avant et après le dépôt respectivement, pour chaque tournée l. Les structures $adj1$ et $adj2$ sont des **repères dans les déplacements** à l'intérieur d'une tournée. En effet, le dépôt est un sommet qui appartient à toutes les tournées. Les tableaux $adj1$ et $adj2$ permettront donc d'identifier les voisins du dépôt dans chaque tournée. De plus, on pourra aussi déterminer la position de chaque sommet d'une tournée grâce à ces structures. Par

exemple, si on suppose que le dépôt est la position i dans la route l, le $i^{ème}$ sommet après $adj2(l)$ y occupe la position $i+2$.
Les listes tabous telles que définies dans [60] restent inchangées. Elles seront mises à jour après chaque diversification, afin d'empêcher les sommets impliqués dans le cross-exchange de retourner dans les routes où ils étaient avant la procédure.

3.3 Choix algorithmiques

3.3.1 Choix du nombre de sommets dans les séquences

3.3.1.1 Dans l'inter-routes

Le nombre de sommets dans chacune des séquences impliquées est sujet à une borne supérieure fixe. De plus, les précautions décritent à la section 3.2 nécessitent que ces paramètres soient déterminés de manière interdépendante comme nous le verrons. Supposons que le cross-exchange est effectué entre deux routes r_1 et r_2. Notons t_i le nombre total de clients dans la route $i = 1, 2$ (excluant le dépôt) et supposons que la séquence S_i se trouve initialement dans la route $i = 1, 2$. Le nombre de sommets dans S_i est noté n_i. Nous considérons que le cross-exchange n'a jamais lieu entre deux tournées qui sont des allers-retour. La première étape du cross-exchange consiste à déterminer la valeur des n_i. Supposons que n_1 est choisi en premier (l'ordre du choix n'est pas important). Alors, n_1 est choisi uniformément entre 0 et t_1. Ensuite, n_2 est déterminé de la manière suivante:

- **si** $n_1 = 0$, n_2 est choisi uniformément entre 2 et $t_2 - 1$, quand r_2 n'est pas un aller-retour. Sinon, n_1 est regénéré uniformément entre 2 et $t_1 - 1$;
- **si** $n_1 = 1$ **et** r_1 **est un aller-retour**, n_2 est choisi uniformément entre 2 et $t_2 - 1$;
- **si** $n_1 = 1$ **et** r_1 **n'est pas un aller-retour**, n_2 est choisi uniformément entre 2 et t_2, quand r_2 n'est pas un aller-retour. Sinon, n_1 est regénéré entre 2 et $t_1 - 1$;

- **si** $n_1 = t_1$ $(t_1 > 1)$, n_2 est choisi uniformément entre 1 et $t_2 - 1$ quand r_2 n'est pas un aller-retour. Sinon, n_1 est regénéré entre 2 et $t_1 - 1$;

- **si** $n_1 \neq t_1$, 1, 0, alors, n_2 peut être choisi librement entre 0 et t_2.

3.3.1.2 Dans l'intra-route

Le nombre de sommets dans chacune des séquences impliquées doit être supérieur ou égal à deux, sinon, l'opération serait un simple mouvement de l'algorithme de base. Ici, les paramètres n_i et S_2 $(i = 1, 2)$ ont la même signification que dans l'inter-routes, à la différence que maintenant, S_1 et S_2 appartiennent à la même route. Supposons que l'intra-route est effectué dans la route r et notons t_r le nombre total de clients dans r(dépôt exclu). Supposons que n_1 est choisi en premier, alors:

- n_1 est choisi uniformément entre 2 et $t_r - 2$, puisque $n_1 + n_2 \leq t_r$ et on doit avoir $n_2 \geq 2$;

- ensuite, n_2 est choisi uniformément entre 2 et $t_r - n_1$.

3.3.2 Choix des séquences

Les paramètres t_i, n_i, S_i, r_i, r et t_r $(i = 1, 2)$ auront la même signification que dans la section précédente.

3.3.2.1 Dans l'inter-routes

Nous décrirons comment les sommets de chacune des séquences sont choisis, sachant que le dépôt n'est jamais déplacé. Nous expliquons le mécanisme pour S_1 et la procédure sera exactement la même pour S_2 puisqu'il s'agit de deux routes distinctes. Sans perdre de généralité, nous supposons que le dépôt se trouve toujours à la position 1. Alors, tous les sommets candidats au cross-exchange sont aux positions 1, 2, ..., $t_1 + 1$. Dénotons p_i la position **dans** r_1, du $i^{ème}$ sommet de S_1 et dénotons v_i, le sommet correspondant. Pour déterminer le premier sommet de S_1, on choisit

aléatoirement sa position - p_1 - entre 2 et $t_1 + 1$. Étant donné que les sommets de chaque séquence doivent avoir des positions contigües dans la route d'où ils viennent, il suffit donc de prendre les $(n_1 - 1)$ successeurs de v_1 pour compléter S_1.
En réalité, la détermination de p_1 est moins triviale. En effet, puisqu'une fois que v_1 est connu, on prend ses successeurs, il faut s'assurer qu'à sa suite, il y a effectivement $(n_1 - 1)$ sommets avant le dépôt. Pour cela, la valeur de p_1 doit laisser assez de marge pour pouvoir compléter la séquence. Donc elle doit être telle que le nombre de sommets après v_1 est supérieur ou égal à $(n_1 - 1)$. Par exemple, la valeur maximale de p_1 est telle qu'après v_1, il reste exactement $n_1 - 1$ sommets avant le dépôt. Donc p_1 sera générée uniformément entre 2 et $(t_1 + 1) - (n_1 - 1)$.

3.3.2.2 Dans l'intra-route

Contrairement à l'inter-routes, les choix des sommets des séquences S_1 et S_2 sont interdépendants dans l'intra-route, puisqu'elles appartiennent à la même route.Nous dénotons $p^j{}_i$, la position dans la route r du $i^{ème}$ sommet de S_j, $j = 1$, 2 et $v^j{}_i$ le sommet correspondant à cette position. Par exemple, $v^1{}_1$ est le premier sommet de S_1 et sa position dans r est $p^1{}_1$. De même, $v^1{}_{n_1}$ est le dernier sommet de S_1 et sa position dans r est $p^1{}_{n_1}$. Nous utilisons une variable aléatoire indicatrice Y pour déterminer si S_2 sera ou non avant de S_1, par rapport au sens de parcours initial de la tournée:

- si $Y = 0$, alors, S_2 sera avant S_1 dans le sens de parcours initial de r. Dans ce cas, il doit y avoir *au moins* $(n_1 + n_2 - 1)$ sommets avant $v^1{}_{n_1}$. S'il y a exactement $(n_1 + n_2 - 1)$ sommets avant $v^1{}_{n_1}$, alors, $v^1{}_{n_1}$ est à la position $(n_1 + n_2 + 1)$ dans r. Donc $(n_1 + n_2 + 1)$ est la plus petite valeur de $p^1{}_{n_1}$. Il en découle que $p^1{}_{n_1}$ sera choisie aléatoirement entre $(n_1 + n_2 + 1)$ et $(t_r + 1)$. Une fois que cette valeur est déterminée, $v^1{}_{n_1}$ est directement obtenu à l'aide de la structure *posit* (voir 3.2). Ensuite, il suffira de prendre les $(n_1 - 1)$ prédécesseurs de ce sommet pour compléter S_1. Une fois que S_1 est obtenu, on sait que la plus grande

valeur possible de ${p^2}_{n_2}$ est $({p^1}_1 - 1)$, c'est-à-dire, quand les deux séquences sont contigües. Étant donné que S_2 est avant S_1 dans le sens de parcours initial de r, ${p^2}_{n_2}$ est choisie aléatoirement entre $(n_2 + 1)$ et $p(1)_1 - 1)$. Puis, on obtient directement ${v^2}_{n_2}$ et il suffira de prendre les $(n_2 - 1)$ prédécesseurs de ce sommet pour compléter S_2;

- si $Y = 1$, alors, S_2 sera après S_1 dans le sens de parcours initial de r. Dans ce cas, il doit y avoir *au moins* $(n_1 + n_2 - 1)$ sommets avant ${v^2}_{n_2}$. S'il y a exactement $(n_1 + n_2 - 1)$ sommets avant ${v^2}_{n_2}$, alors, il est à la position $(n_1 + n_2 + 1)$ dans r. Les séquences sont déterminées de manière analogue au cas précédent, à la différence que les choix seront inversés pour chaque séquence.

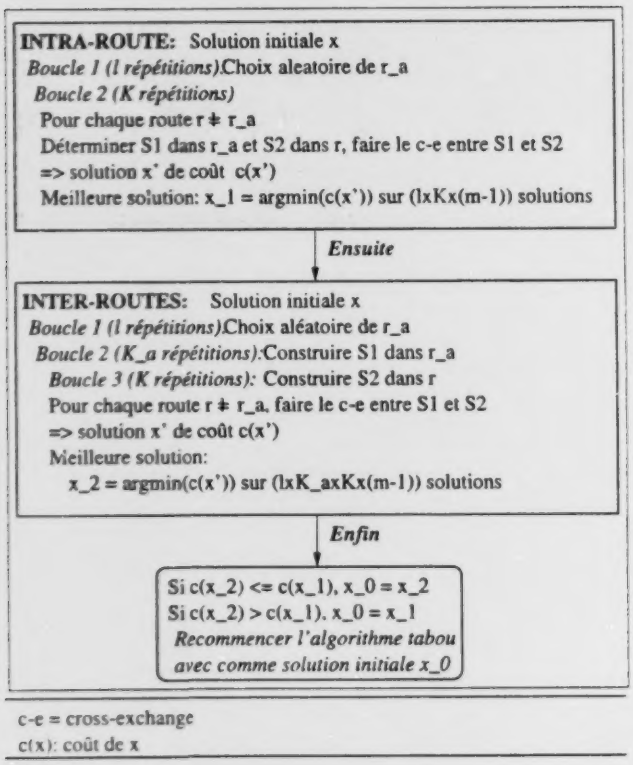
3.3.3 Le déplacement des séquences

Pour déterminer clairement la nouvelle localisation des séquences après le cross-exchange, il faut mettre à jour les identificateurs caractérisant les sommets qu'elles contiennent. Il s'agit des données définies à la section 3.2. Comme le montrent les figures 3.1 et 3.2, le *prédécesseur* du premier sommet de S_1 et le successeur de son dernier sommet doivent être mis à jour dans la route à laquelle appartenait S_2 initialement. La même procédure doit être effectuée pour S_2 dans la route qui contenait S_1 initialement. De plus, si une des séquences contient un sommet qui était voisin du dépôt dans sa tournée initiale, les structures $adj1$ ou $adj2$ devront être mises à jour pour cette tournée. Enfin, la structure *itourn* sera aussi mise à jour pour les sommets de S_1 et S_2 respectivement.

3.4 Mise en oeuvre

Dans cette section, nous expliquons les processus qui permettent d'effectuer l'intra-route et l'inter-routes. Nous dénotons x la solution avant le cross-exchange, x' la solution résultante de cette opération, et m le nombre de tournées dans x. Précisons

Figure 3.8: Algorithme de diversification



que le cross-exchange n'affecte pas m.

3.4.1 Inter-routes

On aurait pu considérer toutes les paires de routes possibles pour faire ce cross-exchange mais cette option occasionnerait des coûts algorithmiques élevés dans les problèmes avec un grand nombre de tournées.

La procédure commence par le choix aléatoire d'une route parmi m, dans x. Notons cette route r_a. Ensuite, r_a est couplée **tour à tour** avec chacune des autres routes de x, afin d'effectuer un cross-exchange, de la manière décrite dans 3.3.3. Pour chaque résultat, le coût total de la solution x' résultante est conservé. Notons qu'alors, $(m-1)$ **solutions différentes seront obtenues**. Supposons que S_1 provient de r_a et que S_2 provient d'une autre route r de x. Il faut comprendre que dans chacune des $(m-1)$ solutions, seulement deux routes – r_a et r – ont changé.

En fait, K_a valeurs différentes de S_1 seront générées dans r_a et K valeurs différentes de S_2 seront générées dans r et ce de manière indépendante, tel que décrit dans 3.3. Pour **chacune des** K_a séquences S_1, un cross-exchange sera effectué avec toutes les K séquences dans r. Par exemple, si nous dénotons ${S_1}^i$ la $i^{ème}$ séquence générée dans r_a et ${S_2}^i$ la $i^{ème}$ séquence générée dans r, on aura des cross-exchanges entre les paires $({S_1}^1, {S_2}^1)$, $({S_1}^1, {S_2}^2)$, ..., $({S_1}^1, {S_2}^K)$, $({S_1}^2, {S_2}^1)$, $({S_1}^2, {S_2}^2)$, ..., $({S_1}^2, {S_2}^K)$, ..., $({S_1}^{K_a}, {S_2}^1)$, $({S_1}^{K_a}, {S_2}^2)$, ..., $({S_1}^{K_a}, {S_2}^K)$. Ainsi, pour chaque paire (r_a, r), on aura un total de $K_a \times K$ cross-exchanges et celui correspondant à une solution de coût total minimal sera retenu.

Pour une route r, nous dénotons r' la route résultante après le cross-exchange et $c(r)$, le coût de r. La construction de x' s'effectue de la manière suivante:

- x' est initialement vide;
- on ajoute ${r_a}'$ dans x';
- parmi toutes les paires de tournées (r_a, r), on détermine celle qui a la propriété

que: $c(r'_a) + c(r') + \sum_{i \neq r_a,\ i \neq r} c(i)$ prend la plus petite valeur.

Alors, r' est incorporée dans x', et les autres tournées sont incorporées dans x' sous leur forme originale dans x.

Le choix aléatoire d'une route permet de ne pas tester toutes les paires de routes possibles. Nous fixons une seule tournée sachant qu'elle sera couplée avec chacune des autres. La paire la plus économique sera finalement retenue.

Précisons qu'on répète l fois le choix aléatoire de r_a et on garde comme solution finale de l'inter-routes la solution x_1^* parmi l dont le coût total est minimal. Ainsi, le nombre total de cross-exchanges effectués est $((K_a \times K) \times (m-1)) \times l$: $(K_a \times K)$ cross-exchanges par paire, $(m-1)$ paires pour une tournée r_a choisie, l choix de r_a. Les données correspondant à x_1^* seront conservées dans une mémoire contenant toutes les informations sur les séquences choisies: leur sens de parcours considéré, les sommets qu'elles contiennent, la valeur de la fonction objectif correspondant à $x^*{}_1$. Cette mémorisation est nécessaire pusiqu'il faudra comparer par la suite $x^*{}_1$ à la solution de l'intra-route.

3.4.2 Intra-route

Le processus de l'intra-route commence par le choix aléatoire d'une tournée parmi m, dans x. Notons r_a cette tournée. Ensuite, l'algorithme détermine les valeurs de n_1, n_2, S_1 et S_2 de la manière décrite dans 3.3. Le cross-exchange tel que décrit dans 3.3.3 est alors effectué dans r_a. Pour chaque route r, nous dénotons r' la route résultante après le cross-exchange.

Après cette première opération, l'intra-route est effectué sur chacune des autres tournées de x. Dénotons $c(i)$ le coût d'une tournée i. En fait, on effectue K cross-exchanges différents par route, c'est-à-dire, on y génère K couples (S_1, S_2) différents. Parmi les K configurations différentes ainsi obtenues pour une route donnée, on choisit celle où le coût de cette route est minimal. La construction de x' s'effectue de la manière suivante:

- x' est initialement vide;

- $r_a{}'$ est incorporée dans x' ;

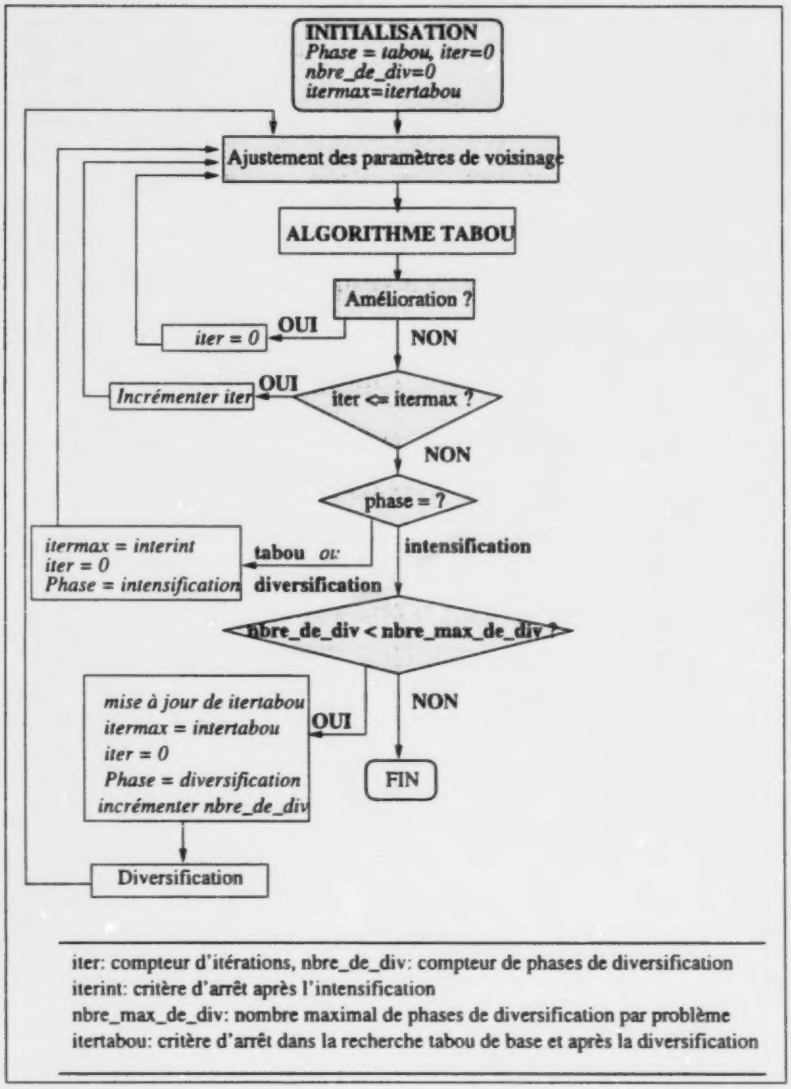
- Pour chaque tournée $i \neq r_a$:
  si $c(i') \leq c(i)$, alors, i' est incorporée dans x'
  si $c(i') > c(i)$, alors, i est incorporée dans x', c'est-à-dire, sous sa forme originale dans x.

Le fait de choisir aléatoirement une tournée, et de l'insérer automatiquement dans x' garantit le changement. En effet, si pour toutes les autres tournées de x, $c(i') > c(i)$, il y a une garantie que x' sera différente de x, par r_a au moins.
Précisons qu'on effectue l choix aléatoires de r_a $(1 < l < m)$, et pour chacun d'eux, le mécanisme décrit ci-dessus sera exécuté. Des l solutions qui en résulteront, on retiendra la solution x_2^* qui aura le coût minimal. Ainsi, un total de $(K \times m) \times l$ cross-exchanges sera effectué pour l'intra-route: K cross-exchanges par tournée, m tournées, l choix aléatoires de r_a.
Comme solution globale de la diversification, on choisit parmi x_1^* et x_2^* celle qui le coût le plus petit. La figure 3.8 illustre l'organisation de l'algorithme de diversification.

3.5 L'algorithme tabou avec diversification

Un simple algorithme tabou provenant de Séguin [60] est décrit au chapitre 2. La procédure de diversification que nous proposons est appliquée dans cet algorithme. L'algorithme tabou avec diversification qui en découle est basé sur quatre étapes principales. Les étapes qui consistent à la recherche dans l'espace des solutions ont une durée limitée. Pour modéliser cela, on utilise un compteur d'itérations *iter*. Au commencement de toute recherche, *iter* est initialisé à 0, puis, incrémenté après l'exécution de chaque mouvement (ou itération). Si le nombre d'itérations (ou de mouvements) sans améliorer la meilleure solution jamais trouvée est supérieur a une

Figure 3.9: Algorithme tabou avec diversification



valeur *itermax* donnée, alors, cette phase de recherche est arrêtée et l'algorithme continue ou se termine. Les quatre phases s'enchaînent de la manière suivante:

Phase 0 La phase d'initialisation de l'algorithme telle que décrite dans 2.2.

Phase 1 La phase de recherche de l'algorithme tabou. Dans cette phase, l'algorithme exécute les mouvements décrits dans 2.2 pour améliorer les solutions. Nous définissons *itertabou* comme le nombre maximal d'itérations sans améliorer la meilleure solution et cette phase s'arrêtera lorsque $iter = itertabou$.

Phase 2 La phase d'intensification telle que décrite dans 2.2. Nous définissons *iterint* comme le nombre maximal d'itérations sans améliorer la meilleure solution et cette phase s'arrêtera lorsque $iter = iterint$. Notons que cette phase est exécutée seulement si l'algorithme tabou a amélioré la solution de départ de la dernière phase de recherche tabou. En effet, dans notre cas, l'intensification est effectuée dans le voisinage de la meilleure solution réalisable trouvée (voir chapitre 2).

Phase 3 La phase de diversification. La solution sur laquelle est appliqué le cross-exchange est la meilleure solution (x^*) trouvée au cours des phases de recherche précédentes. Les mécanismes de la diversification s'enchaînent de la manière suivante:

– **l'initialisation** On initialise la structure *posit* en parcourant chacune des tournées de x^*. Les autres structures sont aussi utilisées dans la recherche tabou et donc sont déjà instanciées;

– **l'exécution de l'intra-route** tel que décrit dans les sections précédentes. Il en résulte une solution $x(intra)$, dont le coût est conservé dans une variable *best_div*;

– **l'exécution de l'inter-routes** tel que décrit dans les sections précédentes. Il en résulte une solution $x(inter)$, dont le coût est comparé à *best_div*. S'il est inférieur ou égal à *best_div*, alors on pose $x_0 = x(inter)$, sinon, on pose

$x_0 = x(intra)$. x_0 sera la solution de départ pour la prochaine phase de recherche dans l'espace des solutions. On incrémente le compteur de phases de diversification exécutées, *nbre_de_div*. Si ce compteur excède une valeur *nbre_max_de_div*, alors, l'algorithme tabou avec diversification s'arrête. *nbre_max_de_div* est un paramètre fourni à l'exécution de l'algorithme.

– **la mise à jour des statuts tabou** des sommets qui étaient dans S_1 et S_2. Si le résultat de la diversification provient de l'intra-route, les sommets de S_1 et S_2 ne devront pas être déplacés dans leur tournée, pendant les t prochaines itérations, où t est la longueur des listes tabou. Si le résultat de la diversification provient de l'inter-routes, les sommets de S_1 et S_2 ne devront pas être déplacés vers leurs tournées originales, avant la diversification, pendant les t prochaines itérations.

– **on recommence l'algorithme tabou avec diversification** (phases 1 à 3), avec comme solution initiale x_0.

La boucle définie par les phases 1 à 3 sera répétée un nombre de fois égal à *nbre_max_de_div*. Par ailleurs, il se peut que les critères d'arrêt pour les phases de recherche changent d'une répétition à l'autre: il suffit de changer la valeur des paramètres *itertabou* ou *iterint*. Cette variation est justifiée par le fait qu'après les premières diversifications, on peut accorder beaucoup de temps à la recherche tabou subséquente. Mais après un grand nombre de diversifications, nous supposons que la meilleure solution est suffisamment bonne, et qu'il sera alors difficile de l'améliorer. Dans ce cas, on n'aimerait pas augmenter l'effort de calcul inutilement et les critères d'arrêt deviennent plus petits. Le diagramme 3.9 montre le cheminement de l'algorithme tabou avec diversification.

Chapitre 4

Résultats

Dans ce chapitre, nous présentons la performance de l'algorithme tabou avec diversification par rapport à la simple recherche tabou. Les deux heuristiques sont comparées sur des bases d'effort de calcul à peu près égaux. Dans un premier temps, nous décrivons les types de problèmes sur lesquels les expériences numériques ont été réalisées. Ensuite, nous définissons les différents critères d'arrêts considérés dans la recherche tabou. Nous analysons la performance de la nouvelle heuristique et nous terminons en proposant d'éventuelles expériences futures afin de l'améliorer. Tous les problèmes ont été résolus sur des stations de travail *Sun Sparc Ultra* 1.

Pour chaque famille de problèmes définie et pour chaque taille, 10 instances de problème seront résolues. La différence d'une instance à l'autre réside dans les valeurs des paramètres stochastiques.

Pour le PTVCDS et le PTVDS, le nombre de routes m, est fixé à 2. Dans l'inter-routes, $K = K_a = 4$, $l = 2$, c'est-à-dire que pour le couplage, on choisit aléatoirement une route 2 fois. Étant donné que $m = 2$, il y a donc 32 cross-exchanges par inter-routes (chapitre 3). Dans l'intra-route, $l = 2$ et $K = 4$, d'où un total de 16 cross-exchanges dans chaque exécution de cette procédure. Ainsi, on aura en tout 48 cross-exchanges par phase de diversification, dans le PTVCDS et le PTVDS. Dans le PVCCS, puisque $m = 1$, seul l'intra-route est appliqué. Dans ce cas, on a pris $K = 8$ et l ne peut prendre que la valeur 1, d'où un total de 8 cross-exchanges.

4.1 Types de problèmes

4.1.1 Le PTVDCS

Dans les problèmes avec demandes et clients stochastiques, nous avons conservé la même classification que dans [60]. En effet, un problème avec n clients est caractérisé par le nombre de clients stochastiques qu'il contient (s) et le coefficient de remplissage des véhicules (f). Ce dernier paramètre exprime la proportion de l'espérance de la demande totale par rapport à la capacité des véhicules.

Pour s et f, trois valeurs ont été choisies respectivement. Ainsi, un problème de taille n peut contenir *1* client stochastique, *(n-1)/2* ou *(n-1)* clients stochastiques, puisque le dépôt est inclus dans n mais est toujours présent. De même, la demande totale espérée correspondra au quart de la capacité des véhicules ($f = 0.25$), à sa moitié ($f = 0.5$) ou aux trois quarts ($f = 0.75$).

En effectuant les différentes combinaisons possibles de s et f, on obtient 9 variantes pour chaque problème de taille n. La notation et la correspondance de celles-ci sont regroupées au tableau 4.1.

Tableau 4.1: Types de problèmes testés dans le PTVCDS lorsqu'il y a n clients

# de clients stochastiques (s)	Coefficients de remplissage des véhicules (f)		
	0.25	0.5	0.75
1	s_1f_1	s_1f_2	s_1f_3
(n-1)/2	s_2f_1	s_2f_2	s_2f_3
(n-1)	s_3f_1	s_3f_2	s_3f_3

La probabilité de présence de chaque client i, i=2,3, ... ,n est uniformément générée dans un intervalle $[\alpha_i, \beta_i]$. Lorsque $s = 1$, $\alpha_i = \beta_i = 0.5$ et pour toute les autres valeurs de ce paramètre, $\alpha_i = 0$ et $\beta_i = 1$.

À chaque client, trois demandes sont assignées. Chacune sera uniformément générée dans les intervalles [1, 9], [5, 15], [10, 20] respectivement. Ainsi, connaissant la loi de probabilité des demandes, la capacité des véhicules sera déduite à partir de la formule

suivante (c.f. [60]):

$$Q = \bar{\epsilon} \sum_{i=2}^{n} (\alpha_i + \beta_i) / 2m\bar{f} \tag{4.1}$$

Q étant la capacité du véhicule, m le nombre de routes et $\bar{\epsilon}$ l'espérance de la demande totale. L'algorithme tabou avec diversification (ATD) et l'algorithme tabou (AT) simple ont été expérimentés pour 19 tailles, entre 6 et 52. Les tableaux de l'annexe A montrent quelles tailles exactement ont été considérées.

4.1.2 Le PVCCS

Pour chaque problème de taille n, on définit deux variantes, chacune étant caractérisée par la valeur de s, le nombre de clients stochastiques qu'elle contient. s n'est pas calculé par une formule fixe comme dans le PTVDCS. Nous notons s^+ la variante contenant le plus grand nombre de clients stochastiques, tandis que s^- désignera la variante contenant le plus petit nombre de clients stochastiques. En moyenne, s^+ contient 1.7 fois plus de clients stochastiques que s^-. Par exemple, pour un problème de taille $n = 10$, il y aura deux sous-problèmes, l'un où le nombre de clients stochastiques est 7 ($s^+ = 7$) et l'autre où le nombre de clients stochastiques est 5 ($s^- = 5$).

En plus, nous définissons 5 familles de problèmes, chacune étant caractérisée par l'intervalle où est générée la probabilité de présence des clients stochastiques. Les 10 variantes ainsi obtenues sont regroupées au tableau 4.2. L'ATD et l'AT simple ont été expérimentés sur des problèmes contenant 10, 20, 30, 40, 50 clients.

4.1.3 Le PTVDS

Dans ce cas, un problème de taille n est caractérisé par la proportion de la demande totale espérée par rapport à la capacité totale des véhicules. Ce paramètre sera

Tableau 4.2: Types de problèmes testés dans le PVCCS lorsqu'il y a n clients

	Intervalle de probabilité de présence des clients ($[a, b]$)				
(s) *parmi* n	$[0.2 - 0.2]$	$[0.5 - 0.5]$	$[0.8 - 0.8]$	$[0.1 - 1]$	$[0.5 - 1]$
plus de clients stochastiques	s_1^+	s_2^+	s_3^+	s_4^+	s_5^+
moins de clients stochastiques	s_1^-	s_2^-	s_3^-	s_4^-	s_5^-

également noté f. Nous avons choisi $f = 0.3$, $f = 0.5$, $f = 0.7$, $f = 0.9$. Chaque problème de taille n aura donc 4 variantes. Nous en donnons la notation au tableau 4.3. Les demandes ont été générées avec l'équation 4.1. Ainsi, l'équation 4.1 a aussi

Tableau 4.3: Types de problèmes testés dans le PTVDS

Coefficients de remplissage (f)	0.3	0.5	0.7	0.9
	f_1	f_2	f_3	f_4

été employée pour déduire la capacité des véhicules. L'ATD et l'AT simple ont été expérimentés sur des problèmes avec 10, 20, 30, 40, 50, 70 clients.

4.2 Critères d'arrêt

Tableau 4.4: Critères d'arrêt pour les deux heuristiques PTVCDS

Notation	Tabou	Intensification	# de diversifications	Diversification
TD1	50×n	100	5	10×n
T2	100×n	100	0	0
TD2	100×n	100	5	20×n
T4	200×n	100	0	0
TD4	200×n	100	5	40×n
T8	400×n	100	0	0
TD8	400×n	100	5	80×n
T16	800×n	100	0	0

Tableau 4.5: Critères d'arrêt pour les deux heuristiques dans le PVCCS

Notation	Tabou	Intensification	# de diversifications	Diversification
TD4	200×n	400	5	40×n
T8	400×n	800	0	0
TD8	400×n	800	5	80×n
T16	800×n	1600	0	0

Tableau 4.6: Critères d'arrêt pour les deux heuristiques dans le PTVDS

Notation	Tabou	Intensification	# de diversifications	Diversification
TD4	200×n	100	5	40×n
T8	400×n	100	0	0

Le critère d'arrêt dans les deux heuristiques est le nombre d'itérations sans améliorer la meilleure solution trouvée jusqu'à présent. Nous avons divisé ce paramètre en différentes composantes, chacune exprimant la condition d'arrêt dans chaque étape de l'algorithme. Ainsi, pour l'algorithme tabou de base, le critère d'arrêt est un couple (t_{tab}, t_{int}). t_{tab} est le critère d'arrêt de la phase générale, t_{int} est celui la recherche tabou après une phase d'intensification (c.f. Chapitre 2).

Dans l'algorithme tabou avec diversification, le triplet $(t_{tabdiv}, t_{intdiv}, div \times t_{div})$ définira la condition d'arrêt. div est le nombre de phases de diversification. t_{tabdiv} est le critère d'arrêt dans la phase générale (de base) de l'algorithme tabou. t_{div} est le critère d'arrêt pour l'algorithme tabou qui suit la phase de diversification et t_{intdiv} est le critère d'arrêt pour l'algorithme tabou qui suit toute phase d'intensification exécutée après la diversification (c.f. Chapitre 3). En général, la valeur de t_{intdiv} dépendra de l'importance accordée à l'intensification après la diversification. Pour notre part, étant donné que l'algorithme est déjà assez avancé à ce niveau, l'intensification n'a pas un critère d'arrêt important. Ainsi, t_{intdiv} sera en général inférieur ou égal à t_{int}. Afin de pouvoir observer l'évolution de la performance de l'algorithme tabou avec di-

versification par rapport à la version de base, nous avons effectué les expériences sur plusieurs échelles de critères d'arrêt. Cela permettra également de tirer quelques conclusions sur le comportement à long terme des deux heuristiques. En déterminant ces différentes échelles, le but est de pouvoir comparer l'algorithme tabou et l'algorithme tabou avec diversification sur des bases de temps plus ou moins identiques. Ainsi, pour une échelle donnée, les critères doivent être tels que:

$$t_{tab} + t_{int} \approx t_{tabdiv} + t_{intdiv} + (div \times t_{div}) \tag{4.2}$$

La première échelle utilisée est la même que dans [60]. Les autres échelles ont été déterminées en doublant chaque fois la plus grande échelle considérée jusque là. Les tableaux 4.4, 4.5 et 4.6 présentent les échelles utilisées pour le PTVCDS, le PVCCS et le PTVDS respectivement. Les heuristiques comparées y sont regroupées 2 à 2. Dans les sections ultérieures, nous utiliserons cette notation abrégée pour désigner les heuristiques.

4.3 Performance de l'algorithme tabou avec diversification

Dans cette section, l'accent sera mis sur les améliorations notables de l'algorithme tabou provoquées par la diversification. Rappelons que le but de cette procédure est surtout d'annuler ou de réduire les grosses erreurs laissées par l'heuristique tabou simple. Nous analysons la performance de la diversification pour les résultats où l'algorithme tabou effectue des erreurs par rapport à la meilleure solution connue. Les moyennes seront cependant calculées par rapport au nombre total de problèmes testés, et non au nombre d'erreurs. Pour l'analyse, nous observerons les réductions des erreurs maximales, des erreurs moyennes et du nombre d'erreurs, obtenues grâce à la diversification.

Tableau 4.7: Erreurs maximales globales du tabou et erreurs correspondantes du tabou+diversification dans le PTVCDS

Méthodes	1			(n-1)/2			(n-1)		
	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
T2	17.54	14.65	11.33	16.92	10.07	25.29	8.67	13.96	6.4
TD1	10.47	8.84	5.09	16.92	9.15	3.06	16.35	0.66	6.4
T4	11.26	10.36	8.73	9.24	4.96	15.12	7.29	5.84	12.34
TD2	0	5.1	4.09	2.22	4.96	3.06	3.06	3.68	2.04
T8	9.71	13.63	7.50	14.00	2.57	3.69	2.64	8.97	5.08
TD4	2.71	6.02	4.09	1.64	3.11	3.69	1.61	0.66	2.06
T16	6.05	9.41	4.57	9.35	1.97	3.69	2.37	2.48	4.18
TD8	5.15	4.64	1.73	7.85	1.15	3.36	0.99	1.57	0.66

4.3.1 Le PTVCDS

Un total de 2710 problèmes ont été testés dans chaque échelle pour le critère d'arrêt. Dans le tableau 4.7, la ligne de l'entrée T2i, $i = 1, 2, 4, 8$ contient l'erreur maximale *absolue* de l'AT simple, sur la totalité des problèmes testés, pour chaque famille de problèmes définie dans le tableau 4.1. Puisqu'il s'agit d'une erreur absolue, l'entrée TDi, $i = 1, 2, 4, 8$ contient l'erreur de l'ATD équivalent (puisque $T2i \approx TDi$), sur *exactement le même problème.* Par exemple, si nous identifions une erreur par le triplet $(s, f, erreur)$, supposons que l'erreur $(1, 0.25, 17.54)$ dans T2 a été faite sur l'instance de problème j, $1 \leq j \leq 10$, d'une taille donnée. Alors, $(1, 0.25, 10.47)$ est l'erreur effectuée par TD1 sur cette même instance de problème, j, pour la même taille. Les erreurs qui sont contenues dans les lignes TDi, $i = 1, 2, 4, 8$ ne sont pas nécessairement les erreurs maximales de l'ATD. Pour cette raison, nous présentons aussi le tableau 4.8, où la ligne TDi contient l'erreur maximale *absolue* de l'ATD pour chaque famille de problème. De manière analogue au tableau 4.7, puisqu'il s'agit d'une erreur absolue, elle représente un problème précis sur une taille précise

Tableau 4.8: Erreurs maximales globales du tabou+diversification et erreurs correspondantes du tabou dans le PTVCDS

Méthodes	1			(n-1)/2			(n-1)		
	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
TD1	10.47	8.84	5.09	16.92	9.15	**4.57**	16.35	**4.05**	**12.64**
T2	17.54	14.65	11.33	16.92	10.07	**8.79**	8.67	**3.00**	**4.35**
TD2	**7.16**	**5.51**	4.09	**16.92**	4.96	**4.57**	3.06	3.68	**6.4**
T4	**7.16**	**9.72**	8.73	**1.72**	4.96	**4.57**	7.29	5.84	**6.4**
TD4	**6.05**	**6.02**	4.09	**3.88**	3.11	**4.47**	**2.76**	**2.12**	2.06
T8	**8.31**	**13.63**	7.5	**5.03**	2.57	**1.49**	**1.99**	**2.48**	5.08
TD8	**5.68**	4.64	**2.06**	7.85	**2.57**	3.36	0.99	1.57	**4.52**
T16	**3.18**	9.41	**4.02**	9.35	**1.89**	3.69	2.37	2.48	**2.39**

et la ligne T2i contient l'erreur de l'AT sur ce même problème. On voit que de nombreuses entrées des tableaux 4.7 et 4.8 sont équivalentes mais qu'il existe quand même des cas (entrées en gras dans le tableau 4.8) où l'erreur maximale de l'ATD ne correspond pas à l'erreur réduite du tableau 4.7.

À partir du tableau 4.7, on remarque que l'ATD réduit l'erreur maximale de l'AT simple 31 fois sur 36, et ce de 30% à 100%. De plus, l'objectif de réduire ou d'annuler les plus grosses erreurs a été atteint. Parmi les plus importantes améliorations, on a:

- des erreurs de **25.29%, 17.54%, 14.65% et 13.96%** qui ont été ramenées par TD1 à **3.06%, 10.47%, 8.84% et 0.66%** respectivement.
- des erreurs de **15.12%, 12.34% et 11.26%** qui ont été ramenées par TD2 à **3.06%, 2.04% et 0%** respectivement
- des erreurs de **13.63% et 14%** qui ont été ramenées par TD4 à **6.02% et 1.64%** respectivement

Les améliorations les plus importantes de l'ATD se situent au niveau des deux premières échelles de critère d'arrêt. Cela est compréhensible dans la mesure où dans T8

et T16, la phase initiale de tabou de base est assez longue, et dans ce cas, la meilleure solution trouvée à la fin de cette phase correspondra à une erreur moins importante ($< 10\%$ en majeure partie). La diversification réduit cette erreur dans la plupart des cas mais l'effet sera moins visible, puisque celle-ci est déjà assez petite.

Dans le tableau 4.8, la performance de l'ATD est supérieure à celle de l'AT 21 fois sur 36. Il existe 6 problèmes où l'AT simple a une performance supérieure à celle de l'ATD et plusieurs cas où les deux heuristiques sont à égalité. Mais ceci peut se justifier par *le sens dans lequel nous considérons l'observation de ces erreurs.* En effet, nous partons de l'erreur maximale de l'ATD, et nous déterminons l'erreur de l'AT simple sur le même problème. Or dans la plupart des problèmes où la performance de l'ATD est inférieure à celle de l'AT (T8 et T16), ce dernier commet déjà de très petites erreurs ($< 4.5\%$) et pour confirmer la remarque précédente, il sera difficile de les améliorer.

Par ailleurs, dans le tableau 4.7, on remarque que TD1 commet l'erreur $(n-1, 0.25, 16.35\%)$, tandis que l'erreur correspondante de T2 est $(n-1, 0.25, 8.67\%)$. De même, dans le tableau 4.8, TD1 commet l'erreur $(n-1, 0.75, 12.64\%)$, tandis que l'erreur correspondante de T2 est $(n-1, 0.75, 4.35\%)$ et TD2 commet l'erreur $((n-1)/2, 0.25, 16.92\%)$, tandis que l'erreur correspondante de T4 est $((n-1)/2,\ 0.75,\ 1.72\%)$. Ce comportement pourrait s'expliquer par le fait que la phase initiale de l'AT de base n'est pas assez longue, puisque dans tous ces cas, la performance de l'ATD reprend le dessus dans l'echelle immédiatement au dessus. En effet, dans le dernier cas par exemple, l'erreur de 16.92% commise par T2 ($T_{tab} = 100 \times n$) passe à 1.72% avec T4 ($T_{tab} = 200 \times n$) et on remarque que pour l'ATD, c'est avec TD4, c'est à dire avec une phase initiale de l'AT de base équivalente ($T_{tabdiv} = 200 \times n$) à celle de T4 qu'on arrive aussi à faire disparaître cette grosse erreur. Cela illustre le fait que la diversification est un atout certain mais le voisinage de base, c'est-à-dire l'AT simple, est lui aussi assez efficace. Trouver **l'équilibre adéquat d'effort de calcul afin d'exploiter au maximum les points forts des deux approches**

n'est pas une tâche triviale. Les tableaux A.3 et A.4 montrent que l'erreur de 16.92% de TD2 a été effectivement réduite à 0.79% par TD4.

Enfin, les résultats de ces heuristiques pour un critère d'arrêt donné, doivent être logiquement meilleurs (au moins égaux) que les résultats pour un critère d'arrêt inférieur. Mais ce n'est pas toujours le cas comme l'illustrent les erreurs $(n-1, 0.75, 6.4\%)$ et $(n-1, 0.75, 12.34\%)$ de T2 et T4 respectivement, dans le tableau 4.7. Cela est tout à fait normal et est dû aux composantes aléatoires utilisées dans la conception de l'AT. En effet, la longueur des listes tabou est une variable aléatoire et le germe utilisé pour leur génération dépend du numéro de l'itération courante. Ainsi, bien que la recherche tabou est plus longue dans T4 par rapport à T2 ($200 \times n$ itérations sans améliorer la meilleure solution dans T4 et $100 \times n$ itérations dans T2), les durées des tabous ne sont pas identiques dans les deux cas – puisque T4 va au-delà de $100 \times n$ itérations sans améliorer la meilleure solution –, d'où la possibilité de différence de performance. Les erreurs maximales par taille des deux algorithmes pour les 4 échelles sont regroupées, dans les tableaux A.1 à A.8.

Le tableau 4.9 présente les erreurs moyennes et le nombre d'erreurs des deux heuristiques en fonction du paramètre s. L'ATD améliore toujours ces deux mesures de performance. Les deux heuristiques semblent avoir une meilleure performance lorsque s augmente. Une étude plus approfondie de la diversification est nécessaire pour justifier ce fait, d'autant plus que dans le PVCCS, le comportement des deux heuristiques est contraire. Cependant dans [60], la même remarque a été observée. Les erreurs moyennes et le nombre d'erreurs par taille pour les deux heuristiques sont regroupées dans les tableaux A.9 à A.16.

Le tableau 4.10 présente la distribution des erreurs des deux heuristiques par intervalle de grandeur. On y trouve également leur nombre total d'erreurs et leur temps moyen total d'exécution (en secondes). Ainsi, TD1 annule entre autres 1 erreur (sur 1) de plus de 25%, 1 erreur (sur 2) de plus de 15%. L'évolution des résultats sur les 4 échelles montre que l'amélioration de performance due à la diversification décroît

Tableau 4.9: Performance des deux algorithmes en fonction de s dans le PTVCDS

Méthodes S	1	(n-1)/2	(n-1)
Erreur moyenne			
TD1	0.53	0.34	0.30
T2	0.85	0.52	0.36
TD2	0.36	0.22	0.17
T4	0.64	0.29	0.25
TD4	0.25	0.12	0.09
T8	0.51	0.22	0.14
TD8	0.22	0.08	0.05
T16	0.38	0.16	0.09
Nombre d'erreurs			
TD1	152	149	190
T2	196	172	215
TD2	136	128	162
T4	164	146	182
TD4	109	98	126
T8	140	126	156
TD8	90	75	95
T16	119	106	132

Tableau 4.10: Erreurs par intervalle pour les deux algorithmes dans le PTVCDS

Intervalle	TD1	T2	TD2	T4	TD4	T8	TD8	T16
[0,1]	315	330	294	306	242	280	196	239
[1,2]	70	104	61	78	56	71	37	62
[2,5]	79	106	64	86	31	55	24	50
[5,10]	21	31	6	17	4	14	3	6
[10,15]	4	8	0	4	0	2	0	0
[15,20]	2	3	1	1	0	0	0	0
[20,25]	0	0	0	0	0	0	0	0
[25,30]	0	1	0	0	0	0	0	0
Nombre total d'erreurs	491	583	426	492	333	422	260	357
Temps moyen total	29.60	35.52	61.12	69.22	118.81	136.29	233.17	275.92

avec le temps d'exécution de la phase tabou initiale. La performance de l'ATD reste cependant supérieure car pour les quatre échelles de temps, il réussit toujours à réduire le nombre d'erreurs (petites et moyennes) laissées par l'heuristique de base, et ce pour des temps de calcul similaires.

4.3.2 Le PVCCS

Un total de 450 problèmes ont été testés. Dans le tableau 4.11, la ligne T2i, $i = 4, 8$ contient l'erreur maximale *absolue* de l'AT pour chaque famille de problèmes, sur la totalité des problèmes testés, pour le critère d'arrêt T2i (tableau 4.5). La ligne TDi contient l'erreur correspondante de l'ATD équivalent ($T2i \approx TDi$). Ici, la correspondance a la même définition que dans la section précédente. Dans le tableau 4.12, la ligne TDi contient l'erreur maximale *absolue* de l'ATD pour chaque famille de problème, pour l'échelle de temps TDi (tableau 4.5). La ligne T2i contient l'erreur correspondante de l'AT, de manière analogue à la section précédente. Dans le tableau 4.11,

Tableau 4.11: Erreurs maximales globales du tabou et erreurs correspondantes du tabou+diversification dans le PVCCS

		Intervalles des probablilités de présence des clients				
	$[a\ b]$	[0.2 0.2]	[0.5 0.5]	[0.8 0.8]	[0.1 1]	[0.5 1]
		Erreur maximale				
s^+	T8	17.01	11.5	17.45	15.59	15.7
	TD4	9.53	4.08	6.98	9.54	7.58
s^-	T8	7.98	13.37	8.12	9.68	7.59
	TD4	0	4.88	6.98	7.41	5.71
s^+	T16	15.12	11.17	16.76	17.69	15.01
	TD8	3.34	3.74	7.45	7.85	7.58
s^-	T16	9.36	7.97	7.96	7.46	12.39
	TD8	2.45	1.13	4.72	5.16	4.62

l'erreur maximale de l'AT est réduite par l'ATD dans tous les cas et ce de manière significative puisqu'on a des réductions allant jusqu'à 60% (il s'agit de l'erreur de

Tableau 4.12: Erreurs maximales globales du tabou+diversification et erreurs correspondantes du tabou dans le PVCCS

		Intervalles des probablilités de présence des clients				
	$[a\ b]$	[0.2 0.2]	[0.5 0.5]	[0.8 0.8]	[0.1 1]	[0.5 1]
		Erreur maximale				
s^+	TD4	9.53	**5.82**	6.98	9.54	7.58
	T8	17.01	**7.7**	17.45	15.59	15.7
s^-	TD4	**4.18**	**4.97**	**7.9**	7.41	5.71
	T8	**6.68**	**6.97**	**8.12**	9.68	7.59
s^+	TD8	5.34	**5.62**	7.45	7.85	7.58
	T16	15.12	**6.52**	16.75	17.69	15.01
s^-	TD8	**3.54**	**3.94**	**5.22**	5.16	**4.87**
	T16	**8.55**	**3.84**	**6.71**	7.46	**6.91**

17.45/ Dans le tableau 4.12, les données en gras présentent les cas où l'erreur maximale de l'ATD ne correspond pas à une réduction de celle de l'AT simple (tableau 4.11). Il semblerait donc que l'AT simple commet de plus grosses erreurs lorsque s est grand, puisque la plupart des situations où l'erreur de l'AT dans ce tableau ne correspond pas à celle du tableau 4.11 correspondent à s^-. De même que dans le PTVCDS, les réductions dues à la diversification sont moins importantes dans ce cas car elles correspondent à des erreurs moins grandes de la recherche tabou simple. Les erreurs maximales par taille des deux heuristiques sont regroupées dans les tableaux B.1 à B.4.

Le tableau 4.13 présente l'erreur moyenne globale et le nombre total d'erreurs des deux heuristiques, pour les 5 familles de problèmes définies. De manière générale, l'erreur moyenne et le nombre d'erreurs sont toujours réduits. *TD4* réduit l'erreur moyenne de 60% à 80% et le nombre d'erreurs de 40% à 60%. TD8 apporte des améliorations dans des proportions similaires. Les tableaux B.5 à B.8 regroupent les erreurs moyennes et le nombre d'erreurs par taille, pour les deux heuristiques.

La performance de la diversification sur les deux échelles de temps considérées ne

Tableau 4.13: Performance des deux algorithmes en fonction de [a, b] dans le PVCCS

[a b]	Intervalles des probablilités de présence des clients [0.2 0.2]	[0.5 0.5]	[0.8 0.8]	[0.1 1]	[0.5 1]
Erreur moyenne					
TD4	0.47	0.43	0.94	0.77	0.65
T8	2.29	1.97	2.71	2.07	2.34
TD8	0.37	0.26	0.65	0.48	0.86
T16	1.58	1.53	2.27	1.83	2.21
Nombre d'erreurs					
TD4	19	20	32	25	22
T8	45	40	50	44	47
TD8	19	15	29	21	26
T16	38	36	48	43	46
Erreurs maximales					
TD4	9.53	4.88	6.98	9.54	7.58
T8	17.01	13.37	17.45	15.59	15.7

réflète pas une différence notable, à part au niveau de l'erreur moyenne, comme on le voit au tableau 4.13. Cela montre que le PVCCS est moins sensible à l'augmentation du critère d'arrêt. Il faudrait donc investir plus dans la qualité du voisinage généré. Par ailleurs, les tableaux 4.11, 4.12 et 4.13 montrent aussi que les problèmes où la probabilité de présence des clients est 0.5 semblent plus faciles à résoudre puisque dans ces cas, les mesures de performance (erreur maximale, erreur moyenne et nombre d'erreurs) sont meilleures. D'un autre côté, les problèmes les plus difficiles semblent être ceux où la probabilité de présence des clients est grande (s_3) ou encore, ceux où ce paramètre peut être très variable (s_4 et s_5). L'effet de la diversification mérite donc plus d'attention dans ces cas. Heureusement, l'erreur maximale y est réduite presque de moitié.

Le tableau 4.14 présente les erreurs des deux heuristiques par intervalle de grandeur. On peut constater une fois de plus l'utilité de la diversification pour améliorer une simple heuristique tabou. En effet, TD4 annule entre autres 6 erreurs sur 6 entre 15% et 20%, 11 erreurs sur 11 entre 10% et 15%. TD8 annule entre autres 5 erreurs sur 5

Tableau 4.14: Erreurs par intervalle pour les deux algorithmes dans le PVCCS

Intervalle	TD4	T8	TD8	T16
[0,1]	41	44	39	49
[1,2]	23	27	29	26
[2,5]	35	66	35	64
[5,10]	19	72	9	62
[10,15]	0	11	0	5
[15,20]	0	6	0	5
[20,25]	0	0	0	0
[25,30]	0	0	0	0
Nombre total d'erreurs	118	226	112	211
Temps moyen total	42.87	32.30	83.18	84.43

entre 15% et 20%, et 5 erreurs sur 5 entre 10% et 15%.

4.3.3 Le PTVDS

Un total de 280 problèmes ont été testés. Malgré que les tests aient été effectués sur une seule échelle de temps, l'effet de la diversification est assez significatif pour conclure à son efficacité.

Le tableau 4.15 présente l'erreur maximale *absolue* de l'AT (ligne T8) sur tous les problèmes testés, ainsi que l'erreur correspondante (voir section 4.3.1) de l'ATD (ligne TD4). De même, le tableau 4.16 présente l'erreur maximale *absolue* de l'ATD (ligne TD4) sur tous les problèmes testés, ainsi que l'erreur correspondante de l'AT (ligne T8). Le tableau 4.17 présente les erreurs moyennes et le nombre d'erreurs des deux heuristiques. On remarque que la diversification a plus d'effet sur les problèmes avec un fort taux de remplissage. Fort heureusement, il s'agit des problèmes où l'heuristique tabou simple commet les plus grosses erreurs. En effet, la réduction de l'erreur maximale, de l'erreur moyenne et du nombre d'erreurs est plus grande dans ces familles de problèmes.

Par exemple, le nombre d'erreurs passe de 37 à 21 lorsque $f = 0.7$ et de 40 à 21

Tableau 4.15: Erreurs maximales globales du tabou et erreurs correspondantes du tabou+diversification dans le PTVDS

f	0.3	0.5	0.7	0.9
T8	10.86	10.74	18.0	10.16
TD4	10.64	10.52	3.47	4.3

Tableau 4.16: Erreurs maximales globales du tabou+diversification et erreurs correspondantes du tabou dans le PTVDS

f	0.3	0.5	0.7	0.9
TD4	10.64	10.52	**4.95**	**4.46**
T8	10.86	10.74	**7.57**	**9.52**

lorsque $f = 0.9$, alors que lorsque $f = 0.7$, il passe de 50 à 45 et lorsque $f = 0.5$, il passe de 49 à 45 seulement. Ce comportement pourrait s'expliquer par le fait que lorsque les demandes sont grandes par rapport à la capacité, il y a plus de ruptures et donc plus de possibilités de parcours dans une route, puisqu'elles sont stochastiques. Une transformation importante dans une solution donnée ouvrirait donc la voie sur un grand nombre de nouvelles configurations possibles. Les erreurs maximales par taille pour les deux heuristiques sont regroupées aux tableaux C.1 et C.2. Les erreurs moyennes et le nombre d'erreurs par taille sont regroupés aux tableaux C.3 et C.4.

Le tableau 4.18 présente la distribution des erreurs des deux aglorithmes pour différents intervalles de grandeur. La diversification aura permi d'annuler la pire erreur, celle qui était supérieure à 15% et 4 autres erreurs plus grandes que 10%.

Notons que dans les tableaux 4.10, 4.14 et 4.18, les nombre d'erreurs comprises dans des intervalles correspondant à des valeurs petites peuvent être plus grand dans la diversification. Cela s'explique par le fait que l'ATD annule ou réduit les grosses erreurs de l'AT simple, à des valeurs plus petites, d'où la forte concentration dans ces intervalles. Cela ne constitue pas un problème important car il s'agit d'erreurs de taille vraiment négligeable et il serait inutile d'essayer de les annuler, d'autant plus

Tableau 4.17: Performance des deux algorithmes en fonction de f dans le PTVDS

f	0.3	0.5	0.7	0.9
Erreur moyenne				
TD4	1.68	1.57	0.48	0.59
T8	2.55	2.24	1.45	1.46
Nombre d'erreurs				
TD4	45	44	21	21
T8	50	49	37	40

que nous étudions des heuristiques.

Tableau 4.18: Erreurs par intervalle pour les deux algorithmes dans le PTVDS

Intervalle	TD4	T8
[0,1]	41	54
[1,2]	31	27
[2,5]	48	61
[5,10]	9	27
[10,15]	2	6
[15,20]	0	1
[20,25]	0	0
[25,30]	0	0
Nombre total d'erreurs	131	176
Temps moyen total	485.32	581.48

4.4 Analyse de la performance de l'algorithme tabou avec diversification

Dans cette section, nous proposons quelques remarques déduites à la lumière des résultats de l'ATD. Dans le PVCCS, les taux de réduction sont très semblables d'un intervalle de probabilité à l'autre. Ainsi, on ne peut pas tirer de conclusion fondée en fonction de ce paramètre. Néanmoins, lorsqu'on observe séparément les problèmes avec moins de clients stochastiques et plus de clients stochastiques, il apparaît que

Tableau 4.19: Erreurs moyennes des deux algorithmes selon le type de problèmes dans le PVCCS

	$[a\ b]$	Intervalles des probablilités de présence des clients [0.2 0.2]	[0.5 0.5]	[0.8 0.8]	[0.1 1]	[0.5 1]
Erreur moyenne						
s^+	TD4	0.69	0.37	0.66	0.85	0.76
	T8	2.75	2.36	3.1	2.29	2.34
s^-	TD4	0.4	0.52	1.28	0.66	0.51
	T8	1.71	1.47	2.21	1.79	1.71
s^+	TD8	0.42	0.35	0.52	0.53	1.06
	T16	1.91	2.03	2.69	2.1	2.51
s^-	TD8	0.31	0.15	0.81	0.42	0.6
	T16	1.16	0.9	1.75	1.48	1.83
Nombre d'erreurs						
s^+	TD4	15	11	14	12	12
	T8	24	25	27	22	25
s^-	TD4	5	9	18	13	10
	T8	21	15	13	22	22
s^+	TD8	9	12	14	11	14
	T16	22	24	25	22	24
s^-	TD8	10	3	15	10	12
	T16	16	12	23	21	22

Tableau 4.20: Performance des deux algorithmes en fonction de f dans le PTVCDS

Méthodes f	0.25	0.5	0.75
Erreur moyenne			
TD1	0.57	0.35	0.27
T2	0.77	0.53	0.43
TD2	0.34	0.23	0.19
T4	0.52	0.32	0.34
TD4	0.20	0.15	0.11
T8	0.39	0.25	0.23
TD8	0.15	0.11	0.08
T16	0.26	0.19	0.17
Nombre d'erreurs			
TD1	198	170	123
T2	228	200	155
TD2	168	141	117
T4	190	162	140
TD4	134	111	88
T8	164	136	122
TD8	104	83	73
T16	128	122	107

ces derniers semblent plus difficiles. En effet, comme on peut le voir aux tableaux 4.11 et 4.19, l'erreur maximale, l'erreur moyenne et le nombre d'erreurs sont presque toujours plus grands lorsqu'il y a plus de clients stochastiques. Cela laisse donc une grande marge d'amélioration par la diversification. Ces tableaux montrent que les pourcentages d'amélioration de la diversification sont plus importants quand il y a plus de clients stochastiques. Dans une autre optique, on pourrait penser de manière raisonnable que comme dans le cas du PTVDS, plus la composante stochastique est importante, plus la diversification est appropriée puisque les solutions deviennent très sensibles au moindre changement. Néanmoins, les tableaux 4.9 et 4.20 pour le PTVDCS démontrent un comportement tout à fait contraire. En effet, les réductions de l'erreur moyenne et du nombre d'erreurs est plus grande lorsque le taux de remplissage est plus grand. De même, la performance de l'ATD est plus visible quand s

est petit. Mais nous savons déjà par [60] que l'heuristique tabou est plus performante sur les problèmes à faible taux de remplissage ou avec peu de clients stochastiques. Cela montre que l'étude de performance du PTVDCS ne peut pas être comparée à celle du PVCCS ou du PTVDS simplement en tenant compte d'une seule des composantes stochastiques du premier problème. En effet, elles semblent interagir et il est impossible de faire un parallèle avec chacun des deux autres types de problèmes. Comme nous le proposons dans les travaux futurs, la diversification devra être étudiée en fonction des types de clients déplacés, afin d'apporter un jugement valable sur son comportement, par rapport aux paramètres du problème.

Par ailleurs, observons la performance de la diversification selon le critère d'arrêt. On sait déjà que les réductions des erreurs maximales sont une fonction décroissante du nombre d'itérations du critère d'arrêt, ce qui n'est pas le cas pour les réductions des erreurs moyennes ou du nombre d'erreurs. Cela s'explique certainement par le fait que l'ATD continue d'annuler ou de réduire les grosses erreurs (voir les tableaux du nombre d'erreurs par intervalles de grandeur) qui n'étaient pas nécessairement des erreurs maximales.

L'observation la plus importante est que dans la résolution des problèmes par l'ATD, il faut laisser juste assez de temps à la phase initiale de tabou de base pour produire des solutions ni trop bonnes, ni trop mauvaises pour que la diversification puisse apporter des améliorations valables. Ce temps de la phase initiale est peut être le paramètre critique dans la performance de l'ATD et sa détermination nécessite des expériences empiriques que nous introduisons dans la section suivante.

4.5 Expérimentations supplémentaires

Dans cette partie, nous présentons quatre versions de l'ATD. Chacune d'elles est caractérisée par le critère d'arrêt aux différentes étapes de l'algorithme (recherche tabou de base, intensification, recherche tabou après la diversification) et le nombre de diversifications. Le but est d'établir le comportement de l'ATD selon le temps

accordé à la recherche tabou avant et après la diversification. De plus, selon le nombre de diversifications effectuées par problème, on aimerait savoir s'il faudrait apporter davantage de changements ou pas. Nous comparons ces quatres versions de l'ATD avec un AT simple pour des efforts de calcul similaires. Les caractéristiques des cinq heuristiques sont regroupées au tableau 4.21. Néanmoins, étant donné que les tests

Tableau 4.21: Critères d'arrêts pour les expérimentations supplémentaires(n clients)

Version	Tabou	Intensification	# de diversifications	Diversification
DIV1	60×n	100	1	30×n
DIV2	40×n	100	3	20×n
DIV3	40×n	100	5	8×n
DIV4	50×n	100	5	10×n
TAB	80×n	100	0	0

Tableau 4.22: Caractéristiques des problèmes de plus grande taille

Taille	# de routes
52	6
76	3
76	8
100	4
100	10

ont été effectués sur des problèmes de grande taille, la solution optimale n'est pas connue dans ces cas. Les comparaisons seront donc faites par rapport à la meilleure solution connue. Seul le cas du PTVDCS a été étudié. Pour chaque type de problème au tableau 4.22, il y a donc les mêmes 9 familles que dans la section précédente, définies selon le nombre de clients stochastiques et le taux de remplissage. Un total de 10 problèmes par famille est généré. Nous avons donc testé 450 problèmes avec les 5 tailles définies dans le tableau 4.22. Cette étude aidera à prédire la performance de la diversification selon le temps accordé à l'AT qu'elle contient. Par ailleurs, elle aidera dans le calibrage des paramètres pour les critères d'arrêts, en tenant compte du nombre de diversifications.

Le tableau 4.23 confirme que la diversification rend l'AT nettement supérieur à sa

version simple. En effet, les quatre algorithmes avec diversification produisent toujours une erreur moyenne et un nombre d'erreurs inférieur à ceux de l'algorithme de base.

Tableau 4.23: Erreurs moyennes [*#erreurs*] totales pour les 5 critères d'arrêts

Version	Erreur moy.[*#d'erreurs*]
DIV1	1.28[292]
DIV2	1.26[286]
DIV3	1.52[281]
DIV4	1.03[258]
TAB	2.01[359]

DIV1 et DIV2 ont des performances similaires et nettement meilleures que DIV3. DIV4 se démarque en supériorité par rapport aux trois autres variantes. Dans la confrontation de DIV1 et DIV2, on pourrait penser que laisser beaucoup de temps à l'AT sans trop diversifier et lui laisser suffisament de temps en apportant plus de phases de diversification sont deux alternatives à peu près équivalentes. Les performances de DIV3 et DIV2 permettent d'apprécier l'importance du temps accordé à l'AT après avoir effectué une diversification. En effet, la phase de recherche tabou initiale a le même critère d'arrêt dans les deux cas. On effectue plus de diversifications dans DIV3 mais l'AT qui suit cette procédure a plus de 2 fois moins de temps que dans DIV2. Ainsi, bien que la diversification est un catalyseur de performance, il est inutile d'en faire usage si on ne laisse pas assez de temps à la recherche tabou qui doit en bénéficier par la suite. Par contre, l'écart de performance entre ces deux versions est peu important par rapport à l'écart entre leurs temps d'exécution respectifs. En effet, à chaque diversification DIV2 a 2,25 fois plus de temps que DIV3, il y a donc de fortes chances que beaucoup de temps soit perdu dans le premier cas. Cette observation est importante dans les contextes de calibrage des critères d'arrêts.
La performance de DIV4 permet de porter un jugement sur le temps accordé à l'AT avant la diversification, le nombre de diversifications et le temps après chaque diver-

Tableau 4.24: Erreurs moyennes [*#erreurs*] totales en fonction de *(s, f)* pour les 5 critères d'arrêts

Version Erreur moy. selon s	1	(n-1)/2	(n-1)
DIV1	1.18[99]	1.59[96]	1.08[97]
DIV2	1.31[102]	1.49[88]	0.97[96]
DIV3	1.21[93]	1.73[98]	1.63[90]
DIV4	0.99[91]	1.09[71]	1.0[86]
TAB	2.29[126]	2.16[117]	1.6[116]
Erreur moy. selon f	**0.25**	**0.5**	**0.75**
DIV1	1.8[98]	1.1[99]	0.9[95]
DIV2	1.6[94]	1.1[99]	1.0[93]
DIV3	2.2[102]	1.3[87]	1.1[92]
DIV4	1.6[79]	0.8[91]	0.7[78]
TAB	2.5[120]	1.9[126]	1.6[113]

sification. Entre DIV4 et DIV3, il y a une différence de 10×n itérations et de 2×n itérations avant et après la diversification respectivement. L'écart de performance est notable cependant. De ce fait, il est raisonnable de penser que l'efficacité de l'ATD dépend de manière très sensible d'un compromis entre le nombre d'itérations à chaque étape. Il faut donner assez de temps à l'AT de base pour générer des solutions de bonne qualité. Ainsi, malgré des changements éventuellement radicaux, on se donne des chances de conserver certains attributs prometteurs dans le résultat final de la diversification. Cependant, la performance de DIV1 indique que cette dernière condition n'est pas suffisante. En effet, le temps de la recherche tabou de base est considérable $(60 \times n)$ mais elle ne produit pas les meilleurs résultats. Il faut donc diversifier suffisamment, tout en donnant assez de temps à la recherche qui suit pour retrouver de meilleures solutions dans l'espace. Les tableaux D.1 à D.5 présentent les performances détaillées des cinq heuristiques.

4.6 Travaux futurs

Dans cette section, nous proposons quelques aspects de la diversification par les cross-exchange que nous jugeons pertinent d'approfondir dans d'éventuels travaux ultérieurs. Nous étudions seulement l'inter-routes.

4.6.1 Expérimentations avec plus d'essais dans la diversification

Tableau 4.25: Caractéristiques des problèmes où plus d'une route est choisie aléatoirement

Taille	# total de routes (m)	# de routes couplées × (m-1)	
76	3	1× 2	3× 2
76	8	1×7	3×7
100	4	1×3	2×3
100	10	1×9	3×9

Comme nous le décrivons au chapitre 3, l'inter-routes s'effectue entre deux routes. Pour un problème avec m routes, une route parmi m est choisie aléatoirement. Ensuite, elle est couplée tour à tour avec chacune des *(m-1)* routes restantes. On obtient donc *(m-1)* solutions différentes et la meilleure sera retenue comme résultat de l'inter-routes. Maintenant, pour augmenter les possibilités d'amélioration, plusieurs routes seront choisies aléatoirement. Par exemple, si l'on décide de faire k choix, on aura un total de $(m-1) \times k$ solutions. La meilleure d'entre elles sera retenue comme résultat de l'inter-routes. Dans cette section, nous présentons quelques résultats préliminaires pour ces deux approches. Les problèmes testés sont de plus grande taille.

Le tableau 4.25 présente les tailles des problèmes testés, ainsi que les types de couplage effectués dans l'inter-routes. Cette expérience a été effectuée dans le cas du PTVDCS seulement. Le nombre de problèmes est déterminé de la même façon qu'à la section précédente donc nous en avons généré un total de 360. Les erreurs moyennes et le nombre d'erreurs sont regroupées au tableau 4.26.

Tableau 4.26: Impact du nombre de routes choisies aléatoirement sur la performance de la diversification: *erreur moyenne par taille* [*# erreurs*] en fonction de $(s,\ f)$

(n-m)	s \ f	1			(n-1)/2			(n-1)		
		0.25	0.50	0.75	0.25	0.50	0.75	0.25	0.50	0.75
	(76-8-1essai(s))	1.72 [8]	0.85 [6]	1.6 [7]	3.81 [7]	0.61 [6]	1.62 [6]	3.72 [8]	1.34 [7]	0.97 [8]
	(76-8-3essai(s))	1.44 [8]	0.5 [5]	1.14 [6]	3.59 [7]	0.34 [5]	0.66 [5]	4.6 [9]	0.8 [8]	0.65 [9]
	(76-3-1essai(s))	1.28 [5]	1.41 [6]	0.62 [5]	1.8 [3]	2.71 [7]	1.78 [8]	3.83 [6]	0.73 [5]	0.48 [7]
	(76-3-3essai(s))	1.07 [5]	0.88 [7]	1.41 [6]	0.98 [3]	1.73 [5]	0.64 [4]	2.3 [6]	1.01 [7]	0.41 [4]
	(100-10-1essai(s))	1.15 [8]	1.34 [7]	1.15 [7]	1.95 [7]	1.08 [4]	0.96 [6]	3.39 [6]	0.45 [7]	0.73 [9]
	(100-10-3essai(s))	1.21 [8]	0.75 [5]	1.82 [7]	2.82 [10]	0.92 [6]	1.23 [7]	2.82 [7]	0.32 [4]	0.62 [9]
	(100-4-1essai(s))	0.92 [6]	0.87 [7]	1.13 [5]	1.39 [6]	1.9 [6]	0.71 [6]	0.39 [4]	0.62 [7]	0.96 [5]
	(100-4-2essai(s))	1.75 [7]	1.01 [8]	1.44 [3]	0.58 [3]	2.14 [7]	0.93 [7]	0.33 [3]	0.69 [8]	0.83 [5]
Erreur moy. glob.	1essai	1.27 [27]	1.12 [25]	1.13 [23]	2.24 [23]	1.58 [22]	1.27 [25]	2.83 [25]	0.79 [27]	0.79 [30]
	>1essai	1.37 [28]	0.79 [25]	1.45 [22]	1.99 [23]	1.28 [23]	0.87 [23]	2.51 [25]	0.71 [27]	0.63 [27]
Erreur moy. glob. par s	1essai	1.17 [75]			1.7 [70]			1.47 [82]		
	>1essai	1.2 [75]			1.38 [69]			1.28 [79]		

Tableau 4.27: Impact du nombre de routes choisies aléatoirement sur la performance de la diversification: *erreur moyenne par taille* [*# erreurs*] en fonction de s

	S	1	(n-1)/2	(n-1)
Erreur moyenne par s	(76-8-1essai(s))	1.39 [21]	2.01 [19]	2.01 [23]
	(76-8-3essai(s))	1.03 [19]	1.53 [17]	2.02 [26]
	(76-3-1essai(s))	1.1 [16]	2.1 [18]	1.68 [18]
	(76-3-3essai(s))	1.12 [18]	1.12 [12]	1.24 [17]
	(100-10-1essai(s))	1.21 [22]	1.33 [17]	1.52 [22]
	(100-10-3essai(s))	1.26 [20]	1.66 [23]	1.25 [20]
	(100-4-1essai(s))	0.97 [18]	1.33 [18]	0.66 [16]
	(100-4-2essai(s))	1.4 [18]	1.22 [17]	0.62 [16]
Erreur Moyenne totale par s	1essai	1.17 [75]	1.7 [70]	1.47 [82]
	>1essai	1.2 [75]	1.38 [69]	1.28 [79]

Lorsqu'un seul essai est effectué par route, l'erreur moyenne totale est de 1.45 et le nombre d'erreurs 227. Pour plus d'essais, le nombre total d'erreurs est de 223 et l'erreur moyenne de 1.3. Bien que la deuxième version semble meilleure que la première, l'écart entre elles n'est pas assez significatif pour conclure.

Si on regarde les résultats par taille au tableau 4.27, seul le problème avec 76 clients et 3 routes semble vraiment donner de meilleurs résultats lorsqu'on fait plus d'essais. Dans les autres cas, la diversification avec plus d'essais est la meilleure approche 2 fois sur 3. Avec 100 clients et 10 routes, elle surpasse la première méthode seulement une fois.

Les résultats obtenus sont donc peu démonstratifs. De prime abord, on pourrait croire que le fait d'évaluer le plus d'essais possibles augmente les chances de trouver une meilleure solution comme résultat du cross-exchange. Néanmoins, cette dernière ne mènera pas nécessairement à de meilleurs résultats une fois dans la recherche tabou. En effet, lorsqu'on choisit aléatoirement une route *une seule fois* dans un

cross-exchange, c'est à dire lorsque $k = 1$, on génère moins de solutions différentes dans ce cross-exchange que si k avait été plus grand - puisqu'on génère $k \times (m-1)$ solutions par cross-exchange. Donc on a moins de choix de solutions, puisque quand $k > 1$, le nombre de solutions générées est plus grand. Cependant, cela n'exclue pas que *par chance*, il se trouve des solutions prometteuses, même parmi un petit nombre de solutions construites à partir d'un cross-exchange. Notons que plus les solutions obtenues sont de bonne qualité, plus il est difficile de les améliorer. Par ailleurs, le degré de transformation occasionné par un cross-exchange est assez significatif pour que d'un résultat à l'autre, la solution obtenue soit dans tous les cas très différente de celle de départ. Les facteurs de détérioration de la fonction objectif correspondante devraient donc être semblables. En conclusion, étant donné que le déroulement d'un cross-exchange nécessite un temps très petit par rapport au temps total de la recherche, nous proposons d'approfondir la politique avec tous les choix aléatoires possibles de routes, à condition que m ne soit pas trop grand. Le nombre total de paires testées sera alors $(m-1) \times m$ contre *(m-1)* dans l'approche originale.

Conclusion

Les résultats présentés au chapitre 4 ont permis d'établir que la recherche tabou (RT) avec diversification appliquée aux problèmes stochastiques de tournées de véhicules confère plus de robustesse à la RT simple. En effet, les expériences effectuées ont permis de résoudre le problème de départ qui était de réduire les grosses erreurs laissées par cette dernière.

L'utilisation de plusieurs échelles de temps dans les critères d'arrêt a permis de dégager des remarques importantes sur le comportement d'une procédure de diversification dans le contexte tabou. Tout d'abord, on a noté qu'à mesure que l'algorithme tabou de base (AT) a un plus grand temps d'exécution, l'ampleur et le nombre de grosses erreurs diminuent. La diversification serait donc utile lorsque le critère d'arrêt de l'AT ne lui permet pas de générer de "trop bonnes" solutions qu'il serait difficile d'améliorer. De fait, les échelles de temps ont montré qu'il valait mieux accorder moins de temps à l'AT, sachant qu'on effectue une diversification, plutôt que lui accorder suffisamment de temps pour trouver de très bonnes solutions sans diversification. En effet, on a vu qu'avec un temps de calcul semblable à celui d'un AT (TDi *vs* $T2i$, $i = 1, 2, 4, 8$). l'algorithme tabou avec diversification (ATD) a une meilleure performance que ce dernier. De plus, on a vu que l'écart de performance rétrécit à mesure que $T2i$ augmente, ce qui confirme le fait que l'effet de la diversification est décroissant avec l'état d'avancement de la RT de base. Cependant, nous avons vu des cas où l'ATD réduit l'erreur maximale de l'AT équivalent (en terme de temps d'exécution) *seulement* à une échelle de temps supérieure. Ces comportements

isolés de l'ATD ont montré que sa performance dépend aussi du fait que la RT de base a été *suffisamment* avancée. En fin de compte, les deux dernières remarques semblent contradictoires, puisque l'une stipule qu'on ne devrait pas accorder trop de temps à la RT de base afin de valider l'effet diversificateur, tandis que l'autre recommande de lui laisser assez de temps pour assurer une bonne performance de la diversification. On peut cependant établir la limite entre elles en clarifiant la signification du terme "*suffisamment*". En effet, un calibrage empirique permettrait d'évaluer quel critère d'arrêt il faut accorder à la RT de base de manière qu'elle soit *assez* avancée pour permettre à la diversification d'améliorer la meilleure solution trouvée, sans être *trop* avancée, sinon la diversification aurait à améliorer une solution de très bonne qualité. Il faut noter qu'ici, une solution de très bonne qualité n'est pas nécessairement la meilleure, seulement, elle sera plus difficile à améliorer. Le calibrage est une tâche difficile qu'on ne saurait généraliser car il s'applique la plupart du temps au problème particulier qui est traité. Nous avons effectué des expériences supplémentaires qu'on peut considérer comme point de départ dans un processus de calibrage, que nous proposons comme sujet de travail futur.

Dans les expériences supplémentaires, le compromis des critères d'arrêts aux différentes étapes de l'ATD apparaît comme un facteur crucial dans sa performance, ce qui corrobore ce que nous avons noté. En effet, la RT *suivant* une diversification doit avoir assez de temps pour retrouver les meilleures solutions dans les parties inexplorées. Cela sera facilité si la première RT est allée suffisamment loin pour incorporer des attributs prometteurs dans ses solutions mais pas assez pour trouver de "trop bonnes" solutions qu'il serait difficile d'améliorer.

D'un autre côté, il demeure quelques cas où la performance de la diversification est peu significative. Dans le PTVDS en particulier, deux erreurs ont été réduites d'un pourcentage plutôt négligeable. Ce comportement apporte l'idée d'étudier l'effet de la diversification de manière qualitative plutôt qu'en termes de réductions des erreurs comme c'est le cas en majeure partie dans ce mémoire. En effet, si nous ex-

périmentions la diversification en fonction des caractéristiques du problème (nombre de clients stochastiques, taux de remplissage), nous serions probablement en mesure de prédire sa performance. Par exemple, on pourrait définir chaque cross-exchange pour un type de sommets bien déterminé. Il serait alors exécuté selon qu'on désire apporter plus de changements ou moins, connaissant l'impact du déplacement de tels sommets dans la structure de la solution courante.

Enfin, plusieurs variantes de l'approche de diversification présentée dans ce mémoire mériteraient d'être étudiées davantage. Une fois encore, le but serait d'approfondir la connaissance du comportement de la diversification et surtout, d'accroître sa performance. À cet effet, on pourrait interchanger dans un *cross-exchange multiple*, un nombre $k > 1$ de paires de segments de sommets, puisque dans la version originale, une seule paire de segments participe à l'échange. Il est raisonnable de penser que l'opérateur résultant aura un effet plus diversificateur. On pourrait en bénéficier des avantages, comme effectuer moins de phases de diversification par problème. Un exemple de cross-exchange multiple est donné à la figure 3.3 où, $k = 3$.

Une autre expérience intéressante serait d'effectuer plusieurs diversifications *de suite* avant de revenir à la RT. L'idée est d'obtenir une diversification plus agressive. Cette approche se distingue de la variante étudiée dans la section 4.6.1 où, on effectue plusieurs cross-exchange de suite **mais chacun sur la solution originale** et on retient celui où la solution résultante est la moins coûteuse. Dans le cas que nous décrivons à présent, plusieurs diversifications sont effectuées les unes à la suite des autres, **chacune étant appliquée sur la solution obtenue de la diversification précédente**. Pour ce faire, lorsqu'on termine la diversification telle que décrite jusqu'ici, au lieu de retourner dans l'algorithme tabou, une autre diversification est effectuée *avec comme entrée, la solution obtenue dans la diversification précédente.* Supposons qu'on désire faire k superpositions et notons la solution à diversifier s_{div}, l'algorithme décrit au chapitre 3 serait modifié comme suit:

1 Algorithme tabou + Intensification

2 Diversification

$i = 1$, s_{div} = meilleure solution trouvée

Tant que $i \leq k$

Effectuer les cross-exchange, le résultat final s_{c-e^*} est la solution correspondant au meilleur d'entre eux

Poser $s_{div} = s_{c-e^*}$, $i = i + 1$

Fin

3 Retour dans l'algorithme tabou

s_{div} est la solution de départ.

Il est néanmoins recommandé de connaître la nature du problème avant d'appliquer ces opérateurs qui semblent plus puissants. Dans les problèmes stochastiques notamment, on pourrait appliquer des opérateurs plus ou moins diversificateurs selon la sensibilité des composantes aléatoires au changement. En effet, nous avons vu que les problèmes avec plus de clients stochastiques ou avec un fort taux de remplissage semblent plus sensibles à la diversification. Cela rejoint la proposition que nous avons faite ci-dessus, à savoir que l'étude approfondie de la diversification nécessitera l'analyse paramétrique (ou qualitative) des problèmes étudiés.

Bibliographie

[1] A.S. Alfa, S.S. Heragu, et M. Chen. A 3-opt based simulated annealing algorithm for the vehicle routing problems. *Computer and Industrial Engineering*, 21:635–639, 1991.

[2] V.A. Armentano et D.P. Ronconi. Tabu search solution for total tardiness minimization in flowshop scheduling problem. *Computers and Operations Research*, 26:219–235, 1999.

[3] G. Barbarosoglu et D. Ozgur. A tabu search algorithm for the vehicle routing problem. *Computers and Operations Research*, 26:255–270, 1998.

[4] J.F. Bard, L. Huang, M. Dror, et P. Jaillet. A branch and cut algorithm for the VRP with satellite facilities. *IEE Transactions*, 30:821–834, 1998.

[5] J.Jr. Bartholdi, L.K. Platzman, R.L Collins, et W.H. Warden. A minimal technology routing system for meals on wheels. *Interfaces*, 13:1–8, 1983.

[6] C.B. Basnet, D.N. Clark, et L.R. Foulds. A decision support system approach to milk tanker routing. *Control and Cybernetics*, 22:131–144, 1993.

[7] D. Berger. Rsolution d'un problme de conception de rseaux l'aide d'une recherche avec tabous. Rapport technique 98-15, Centre de recherche sur les transports, Université de Montréal, 1998.

[8] D. Bertsimas et L.H Howell. Further results on the probabilistic traveling salesman problem. *European Journal of Operations Research*, 65:68–95, 1993.

[9] D.J. Bertsimas, P. Jaillet, et A.R. Odoni. A priori optimization. *Operations Research*, 38:1019–1033, 1990.

[10] B. Bouzaiene-Ayari, M. Dror, et G. Laporte. Vehicle routing with stochastic demands and split deliveries. *Foundations of Computing and Decision Sciences*, 18:63–69, 1993.

[11] J. Braca, J. Bramel, B. Posner, et D. Simchi-Levi. A computerized approach to the New York City school bus routing problem. *IIEE Transactions*, 29:693–702, 1997.

[12] T.G. Crainic, M. Gendreau, et J.M. Farvolden. A simplex-based tabu search method for capacitated network design. Rapport technique CRT 98-37, Centre de recherche sur les transports, Université de Montréal., 1998.

[13] G. Dantzig et H. Ramser. The truck dispatching problem. *Management Sciences*, 6:81–91, 1959.

[14] G.B. Dantzig, D.R. Fulkerson, et S.M. Johnson. Solution of a large scale traveling saleman problem. *Operations Research*, 2:392–410, 1954.

[15] M. Dror, G. Laporte, et P. Trudeau. Vehicle routing with stochastic demands: Properties and solutions frameworks. *Transportation Science*, 23:166–176, 1989.

[16] R.W. Eglese. Routing winter gritting vehicles. *Discrete Applied Mathematics*, 48:231–244, 1994.

[17] C. Friden, A. Hertz, et D. De Werra. STABULUS: A technique for finding stable sets in large graphs with tabu search. *Computing*, 42:35–44, 1989.

[18] M. Gendreau, A. Hertz, et G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management Sciences*, 40:1276–1290, 1994.

[19] M. Gendreau, G. Laporte, C. Musaraganyi, et E.D. Taillard. A tabu search heuristic for the heterogeneous fleet vehicle routing problem. *Computers and Operations Research*, 26:1153–1173, 1999.

[20] M. Gendreau, G. Laporte, et J.-Y. Potvin. *Vehicle Routing: Modern heuristics.* E. Aarts and J.K. Lenstra, editeurs, Local search in combinatorial optimization, J. Wiley & Sons Ltd., London, England, 1997.

[21] M. Gendreau, G. Laporte, et R. Séguin. An exact algorithm for the vehicle routing problem with stochastic demands and customers. *Transportation Science*, 29:143–155, 1995.

[22] M. Gendreau, G. Laporte, et R. Séguin. Stochastic vehicle routing. *European Journal of Operational Research*, 88:3–12, 1996.

[23] R. Gill et C.S. Syan. An integrated reactive tabu search for flowshop machine sequencing problems. *In Poceedings of the Twelfth International Conference on CAD/CAM Robotics and Factories of the Future*, pages 621–626, 1997.

[24] F. Glover. Candidate list strategies and tabu search. Rapport technique, University of Colorado, Boulder, 1989.

[25] F. Glover. Tabu search, part I. *ORSA Journal on Computing*, 1:190–206, 1989.

[26] F. Glover. Tabu search, part II. *ORSA Journal on Computing*, 2:4–32, 1990.

[27] F. Glover et M. Laguna. *Tabu Search.* Boston: Kluwer Academics Publishers, 1997.

[28] F. Glover, E. Taillard, et D. De Werra. A user's guide to tabu search. *Annals of Operation Research*, 41:3–28, 1993.

[29] S. Hanafi, F. Amrani, et A. Freville. Solving large-scale 0-1 integer programming models with tabu search. In *First International Conference on Operations and Quantitative Management*, pages 32–39, 1997.

[30] A. Hertz, G. Laporte, et M. Mittaz. A tabu search heuristic for the capacitated arc routing problem. Rapport technique CRT 97-03, Centre de recherche sur les transports, Université de Montréal, 1997.

[31] R. Hubscher et F. Glover. Applying tabu search with influential diversifications strategies to multiprocessors scheduling. *Computers and Operations Research*, 21:877–884, 1994.

[32] P. Jaillet. A priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Operations Research*, 36:929–936, 1988.

[33] R. Kwan, Z. Liping, R. Clement, et M. Rahin. Applications of information technology for bus and driver scheduling. Rapport technique, Division of Operational Research and Information Systems, University of Leeds, 1993.

[34] M. Laguna, J.W. Barnes, et F. Glover. Tabu search methods for a single machine scheduling problem. *Journal of Intelligent Manufacturing*, 2:63–74, 1991.

[35] M. Laguna, R. Marti, et V. Campos. Intensification and diversification with elite tabu search solution for the linear ordering problem. *Computers and Operations Research*, 26:1217–1230, 1998.

[36] V. Lambert, G. Laporte, et F.V. Louveaux. Designing collection routes through bank branches. *Computers and Operations Research*, 20:783–791, 1993.

[37] G. Laporte, F.V. Louveaux, et H. Mercure. A priori optimization of the probabilistic traveling salesman problem. *Operations Research*, 42:543–549, 1994.

[38] G. Laporte, F.V. Louveaux, et L. Van Hamme. Integer L-shaped algorithm for the capacited vehicle routing with stochastic demands. Rapport technique CRT 98-39, Centre de recherche sur les transports, Université de Montréal, 1998.

[39] G. Laporte et I.H. Osman. Routing problems: A bibliography. *Annals of Operations Research*, pages 227–262, 1995.

[40] G. Laporte, J.-Y. Potvin, et F. Quilleret. A tabu search heuristic using genetic diversification for the clustered traveling salesman problem. *Journal of Heuristics*, 3:187–200, 1997.

[41] C.Y. Lee et S.J. Koh. A design of the minimum cost ring chain network with dual homing survivability: A tabu search approach. *Computers and Operations Research*, 24:883–897, 1997.

[42] A.N. Letchford et R.W Eglese. The rural postman problem with deadline classes. *European Journal of Operational Research*, 105:390–400, 1998.

[43] S. Lin et K. Kernighan. An effective heuristic for the traveling salesman problem. *Operations Research*, 21:498–516, 1973.

[44] A. Lokketangen et F. Glover. Solving zero-one mixed integer programming problems using tabu search. *European Journal of Operational Research*, 106:624–658, 1998.

[45] F. Maffioli et A. Sciomachen. Ordering problems with side constraints. *Annals of Operations Research*, 69:277–297, 1997.

[46] S. Minghe. A tabu search procedure for solving transportation problem with exclusionnary side constraints. *Journal of Heuristics*, 3:305–326, 1998.

[47] R.J. Mitchell, B. Chambers, et A.P. Anderson. Array pattern synthesis in the complex plane optimised by a genetic algorithm. *Electronics Letters*, 32:10843–1845, 1996.

[48] N. Mladenovic et P. Hansen. Variable neighbourhood search. *Computers and Operations Research*, 24:1097–10100, 1997.

[49] H.L. Ong, B.W. Ang, T.N. Goh, et C.C. Deng. A vehicle routing and scheduling problem with time windows and stochastic demand constraints. *Asia-Pacific Journal of Operational Research*, 14:1–17, 1997.

[50] I. Or. *Traveling Salesman Type Combinatorial Problems and Their Relation to the Logistics of Blood Banking.* Thèse de doctorat, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, 1976.

[51] I.H. Osman. Metastrategy simulated annealing and tabu search algorithm for the vehicle routing problem. *Annals of Operations Research*, 41:421–451, 1993.

[52] I.H. Osman. A tabu search algorithm for the vehicle routing problem. *Management Sciences*, 40:1276–1290, 1994.

[53] I.H. Osman. Heuristics for the generalised assignment problem: Simulated annealing and tabu search approaches. *Operations Research Spektrum*, 17:211–225, 1995.

[54] I.H. Osman et G. Laporte. Metaheuristics: a bibliography. *Annals of Operations Research*, 63:513–623, 1996.

[55] G. Pesant, M. Gendreau, J.-Y. Potvin, et J.-M. Rousseau. An exact constraint logic programming algorithm for the traveling salesman problem with time windows. *Transportation Science*, 32:12–29, 1998.

[56] H. Psaraftis. A k-interchange procedure for local search in precedence constrained vehicle routing problem. *European Journal of Operations Research*, 13:391–402, 1983.

[57] Y. Rochat et E. Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1:147–167, 1995.

[58] E. Rolland, D.A. Shilling, et J.R. Current. An efficient tabu search procedure for the p-median problem. *European Journal of Operations Research*, 96:329–342, 1997.

[59] C.N. Schieffer. A parallel tabu search for large scale traveling salesman problem. *Discrete Applied Mathematics*, 51:243–267, 1994.

[60] R. Séguin. *Problemes stochastiques de tournées de véhicules.* Thèse de doctorat, Université de Montréal, département d'informatique et de recherche opérationnelle, 1994.

[61] J. Shakaprani et J. Skorin-Kapov. Massively parallel tabu search for the quadratic assignment problem. *Annals of Operations Research*, 41:327–341, 1993.

[62] J. Song et T. Lee. A tabu search heuristic for periodic job shop scheduling. *Computer and Industrial Engineering*, 30:433–437, 1996.

[63] P. Soriano et M. Gendreau. Diversification strategies in tabu search algorithms for the maximum clique problem. *Annals of Operations Research*, 63:189–207, 1996.

[64] P. Soriano et M. Gendreau. Fondements et applications des méthodes de recherche avec tabous. *RAIRO Recherche opérationnelle/Operations Research*, 31:133–159, 1997.

[65] E. Taillard. Robust taboo search for the quadratic assignment problem. *Parallel Computing*, 17:443–455, 1991.

[66] E. Taillard, P. Badeau, M. Gendreau, F. Guertin, et J.-Y. Potvin. A new neighborhood structure for the vehicle routing problem with time windows. Rapport technique CRT 95-66, Centre de recherche sur les transports, Université de Montréal, 1995.

[67] E. Taillard, P. Badeau, M. Gendreau, F. Guertin, et J.-Y. Potvin. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31:170–186, 1997.

[68] P. Thompson et H. Psarafatis. Cyclic transfer algorithms for multivehicle routing and scheduling problems. *Operation Research*, 41:935–946, 1993.

[69] Tillman. The multiple terminal delivery problem with probabilistic demands. *Transportation Science*, 3:192–204, 1969.

[70] P. Trudeau, J.-M. Rousseau, J.A. Ferland, et J. Choquette. An operations research approach for the planning and operation of an ambulance service. *INFOR*, 27:95–113, 1989.

[71] D.L. Woddruff et M.L. Spearman. Sequencing and batching for two classes of jobs with deadlines and setup times. *Production and Operation Management*, 1:87–102, 1992.

[72] D.L. Woodruff et M.L. Spearman. A framework for diversification in tabu search. Rapport technique, Department of Industrial Engineering and Management Sciences, Northwestern University, 1990.

[73] J. Xu et J.P. Kelly. A network flow-based tabu search heuristic for the vehicle routing problem. *Transportation Science*, 30:379–393, 1996.

[74] M. Yagiura et T. Ibaraki. Metaheuristics as robust and simple optimization tools. *In Proceedings of 1996 IEEE International Conference on Evolutionnary Computation*, pages 541–546, 1996.

Annexe A

Erreurs maximales et erreurs moyennes par taille dans le PTVDCS

Les tableaux de cet annexe se suivent par paire (TDi, T2i), $i = 1, 2, 4, 8$ pour chaque mesure de performance (erreur maximale et erreur moyenne).

Les tableaux A.2, A.4, A.6 et A.8, regroupent les erreurs maximales *absolues* par taille de l'algorithme tabou (AT) simple pour les 9 familles de problèmes définies dans le PTVDCS, pour les versions T2, T4, T8 et T16 respectivement. Les tableaux A.1, A.3, A.5 et A.7 présentent les erreurs maximales *absolues* par taille de l'algorithme tabou avec diversification (ATD) pour les 9 familles de problèmes définies dans le PTVDCS, pour les versions TD1, TD2, TD4 et TD8 respectivement.

De même, les tableaux A.10, A.12, A.14 et A.16, regroupent les erreurs moyennes par taille de l'AT simple, pour les 9 familles de problèmes définies dans le PTVDCS, pour les versions T2, T4, T8 et T16 respectivement. Les tableaux A.9, A.11, A.13 et A.15 présentent les erreurs moyennes par taille de l'ATD, pour les 9 familles de problèmes définies dans le PTVDCS, pour les versions TD1, TD2, TD4 et TD8 respectivement.

Tableau A.1: Erreur maximale pour TD1 dans le PTVCDS

n \ s / f	1			(n-1)/2			(n-1)		
	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
6	*	*	*	*	*	*	*	*	*
7	*	*	*	*	*	*	1.01	*	*
8	*	*	*	*	0.05	*	*	*	*
9	2.28	*	*	*	*	*	*	*	*
10	*	*	*	1.28	0.38	*	*	1.12	0.01
11	0.32	*	*	*	0.06	0.03	*	*	0.09
12	2.50	*	*	1.19	*	*	0.58	0.65	0.09
13	2.16	0.46	0.21	0.16	*	*	0.21	0.19	*
14	0.63	0.44	0.63	3.21	0.38	*	2.26	1.68	0.26
15	0.02	0.16	0.93	1.56	0.22	*	0.74	0.65	0.09
16	0.79	0.30	*	0.79	0.88	0.15	1.00	0.77	0.72
18	1.33	1.37	0.37	0.21	0.37	2.05	0.10	0.20	0.37
21	4.03	7.31	0.28	0.29	1.28	*	0.24	0.61	3.35
22	3.21	3.21	1.29	16.92	0.68	2.67	1.08	0.66	12.64
28	4.12	4.28	3.65	2.28	1.47	0.74	2.28	0.74	0.38
34	3.26	6.54	4.35	2.39	2.61	0.23	2.39	0.23	1.76
40	6.75	6.62	3.73	3.16	3.10	4.57	1.49	1.50	2.37
46	10.47	8.84	5.09	9.41	9.15	3.06	16.35	1.61	3.65
52	8.49	6.68	4.02	10.11	4.96	3.89	14.36	4.05	6.40
Er. max. globale	10.47	8.84	5.09	16.92	9.15	4.57	16.35	4.05	12.64
Er. max. globale par s	*1*			*(n-1)/2*			*(n-1)*		
	10.47			16.92			16.35		
Er. max. globale par f	*0.25*			*0.5*			*0.75*		
	16.92			9.15			12.64		
Temps moy. tot.	18.26	26.71	18.54	15.05	42.05	30.37	29.66	46.85	38.91

Tableau A.2: Erreur maximale pour T2 dans le PTVCDS

n \ s	1			(n-1)/2			(n-1)		
f	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
6	*	*	*	*	*	*	*	*	*
7	*	*	*	*	*	*	0.03	*	*
8	*	*	*	*	0.05	*	*	*	*
9	*	*	*	*	*	*	*	*	0.13
10	*	*	*	2.04	*	0.12	*	1.12	*
11	*	0.66	*	0.72	0.06	0.07	0.37	*	0.09
12	2.50	1.78	*	1.19	*	*	1.10	0.50	0.33
13	2.16	0.46	0.21	0.95	*	*	0.21	*	*
14	2.36	1.02	0.45	4.23	0.38	0.51	2.01	1.88	*
15	0.11	0.17	0.93	1.82	*	0.10	0.74	0.50	0.33
16	1.86	4.31	*	3.03	1.26	0.26	0.89	0.77	0.72
18	4.38	0.97	0.11	0.21	0.61	*	0.56	0.20	1.40
21	4.71	7.31	*	1.13	0.93	0.01	1.53	1.02	2.75
22	4.12	3.96	1.29	16.92	0.68	2.67	2.46	13.96	4.35
28	7.16	7.17	4.97	2.28	2.29	0.52	2.28	0.52	1.01
34	3.84	4.82	3.96	2.95	3.00	1.87	2.95	1.87	1.77
40	10.53	2.51	4.93	3.07	8.23	8.79	1.57	1.50	2.37
46	17.54	14.65	11.33	16.08	10.07	25.29	8.67	11.21	4.15
52	10.45	10.42	6.48	5.78	6.98	3.69	7.29	3.00	6.40
Er. max. globale	17.54	14.65	11.33	16.92	10.07	25.29	8.67	13.96	6.40
Er. max. globale par s		*1*			*(n-1)/2*			*(n-1)*	
		17.54			25.29			13.96	
Er. max. globale par f		*0.25*			*0.5*			*0.75*	
		17.54			14.65			25.29	
Temps moy. tot.	18.63	22.08	22.60	17.08	51.09	40.33	40.21	57.75	49.90

Tableau A.3: Erreur maximale pour TD2 dans le PTVCDS

n \ s	1			(n-1)/2			(n-1)		
f	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
6	*	*	*	*	*	*	*	*	*
7	*	*	*	*	*	*	*	*	*
8	*	*	*	*	*	*	*	*	*
9	*	*	*	*	*	*	*	*	0.13
10	*	*	*	1.28	*	0.12	*	1.12	*
11	*	0.66	*	0.72	0.06	0.03	*	*	0.09
12	0.36	*	*	*	*	*	0.53	0.50	0.09
13	0.46	0.46	0.21	*	*	*	0.21	*	*
14	0.11	0.27	0.45	0.12	0.38	0.25	2.01	*	*
15	0.02	0.16	*	1.20	*	*	0.21	0.50	0.09
16	0.79	2.06	*	0.79	0.39	0.26	0.80	0.77	*
18	1.28	0.97	0.11	0.21	0.02	*	0.06	0.19	1.40
21	2.39	4.39	*	0.48	0.93	*	0.38	0.96	0.86
22	4.12	3.96	1.29	16.92	0.53	2.67	2.46	0.91	4.32
28	7.16	4.28	2.00	2.28	1.89	0.50	2.28	0.50	1.01
34	2.20	0.91	3.96	2.95	1.05	1.29	2.95	1.29	1.02
40	5.00	2.00	2.87	2.89	3.09	4.57	1.57	1.50	2.30
46	4.86	5.10	4.09	2.22	1.61	3.06	0.90	3.68	2.04
52	4.94	5.51	4.02	4.92	4.96	1.64	3.06	2.12	6.40
Er. max. globale	7.16	5.51	4.09	16.92	4.96	4.57	3.06	3.68	6.40
Er. max. globale par s	*1*			*(n-1)/2*			*(n-1)*		
	7.16			16.92			6.40		
Er. max. globale par f	*0.25*			*0.5*			*0.75*		
	16.92			5.51			6.40		
Temps moy. tot.	35.66	53.66	37.84	29.55	87.37	63.61	67.26	93.81	81.33

Tableau A.4: Erreur maximale pour T4 dans le PTVCDS

n \ s	1			(n-1)/2			(n-1)		
f	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
6	*	*	*	*	*	*	*	*	*
7	11.26	*	*	*	*	*	*	*	*
8	*	*	*	*	*	*	*	*	*
9	*	*	*	*	*	*	*	*	*
10	*	*	*	*	*	*	*	*	*
11	*	*	*	*	0.06	0.07	*	*	0.09
12	0.36	*	*	1.19	*	*	0.97	0.50	0.33
13	0.43	0.06	*	*	*	*	0.21	*	*
14	0.62	0.21	*	0.42	*	0.51	2.01	0.87	*
15	*	2.11	0.93	1.33	*	*	0.74	0.50	0.33
16	0.97	2.06	*	0.94	0.55	0.26	0.80	0.13	0.11
18	4.38	0.78	0.11	0.21	0.02	*	*	0.03	1.40
21	4.71	0.45	*	1.13	0.93	*	1.39	0.28	1.58
22	4.12	3.96	1.29	1.72	0.68	0.13	0.50	0.79	0.09
28	7.16	7.17	4.97	2.81	1.97	0.75	2.81	0.75	1.01
34	3.75	4.82	4.13	2.49	1.53	1.29	2.49	1.29	1.76
40	10.53	2.51	4.93	3.07	2.40	4.57	2.64	1.42	2.37
46	7.92	10.36	8.73	9.24	4.62	15.12	5.82	5.84	12.34
52	7.60	9.72	4.02	4.94	4.96	3.69	7.29	2.48	6.40
Er. max. globale	11.26	10.36	8.73	9.24	4.96	15.12	7.29	5.84	12.34
Er. max. globale par s		*1*			*(n-1)/2*			*(n-1)*	
		11.26			15.12			12.34	
Er. max. globale par f		*0.25*			*0.5*			*0.75*	
		11.26			10.36			15.12	
Temps moy. tot.	40.56	43.23	46.14	34.61	106.62	80.69	79.92	103.98	87.26

Tableau A.5: Erreur maximale pour TD4 dans le PTVCDS

n \ s	1			(n-1)/2			(n-1)		
f	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
6	*	*	*	*	*	*	*	*	*
7	*	*	*	*	*	*	*	*	*
8	*	*	*	*	*	*	*	*	*
9	*	*	*	*	*	*	*	*	*
10	*	*	*	*	*	*	*	*	*
11	*	*	*	*	*	0.07	*	*	0.09
12	0.36	*	*	1.19	*	*	*	0.41	0.09
13	*	*	*	*	*	*	0.21	*	*
14	*	0.02	*	*	*	0.51	2.01	0.87	*
15	*	0.16	*	*	*	*	0.07	0.41	0.09
16	0.79	0.12	*	0.94	0.55	0.15	0.19	0.10	0.11
18	1.71	0.65	*	0.21	*	*	*	*	*
21	1.13	0.45	*	1.13	0.93	*	0.30	0.22	0.36
22	3.21	3.96	0.90	0.79	0.26	0.10	0.37	0.79	0.09
28	4.01	1.79	1.42	1.42	1.39	0.57	1.42	0.57	1.01
34	2.60	1.50	2.47	2.35	0.82	1.29	2.35	1.29	0.44
40	2.71	2.51	2.00	3.07	1.26	4.47	1.61	0.41	0.05
46	6.05	6.02	4.09	1.64	1.19	1.46	0.24	0.66	0.97
52	4.21	5.53	4.02	3.88	3.11	3.69	2.76	2.12	2.06
Er. max. globale	6.05	6.02	4.09	3.88	3.11	4.47	2.76	2.12	2.06
Er. max. globale par s	*1*			*(n-1)/2*			*(n-1)*		
	6.05			4.47			2.76		
Er. max. globale par f	*0.25*			*0.5*			*0.75*		
	6.05			6.02			4.47		
Temps moy. tot.	74.71	104.15	77.37	59.31	174.03	125.88	129.35	173.30	151.16

Tableau A.6: Erreur maximale pour T8 dans le PTVCDS

n \ s	1			(n-1)/2			(n-1)		
f	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
6	*	*	*	*	*	*	*	*	*
7	*	*	*	*	*	*	*	*	*
8	*	*	*	*	*	*	*	*	*
9	*	*	*	*	*	*	*	*	*
10	*	*	*	*	*	*	*	*	*
11	*	*	*	*	*	*	*	*	0.09
12	0.26	*	*	0.93	*	*	0.30	*	0.33
13	0.46	*	*	*	*	*	0.21	*	*
14	0.11	0.27	*	0.42	*	*	0.59	0.62	*
15	*	1.95	*	*	*	0.04	0.07	*	0.33
16	0.97	4.31	*	0.94	*	0.26	0.80	0.10	0.11
18	4.38	0.65	*	*	0.02	*	0.02	0.14	0.16
21	4.71	0.24	*	1.13	0.66	*	0.80	0.52	0.94
22	4.12	2.97	1.29	0.72	0.53	0.13	0.50	0.79	0.09
28	2.43	4.28	2.00	1.04	1.97	1.23	1.04	1.23	0.64
34	3.75	3.66	3.24	2.49	1.53	0.19	2.49	0.19	1.77
40	9.71	2.51	4.95	8.38	2.43	1.49	2.64	0.38	1.41
46	8.31	13.63	7.50	14.00	1.19	3.09	1.48	8.97	3.26
52	7.60	9.75	4.02	5.03	2.57	3.69	1.99	2.48	5.08
Er. max. globale	9.71	13.63	7.50	14.00	2.57	3.69	2.64	8.97	5.08
Er. max. globale par s		*1*			*(n-1)/2*			*(n-1)*	
		13.63			14.00			8.97	
Er. max. globale par f		*0.25*			*0.5*			*0.75*	
		14.00			13.63			7.50	
Temps moy. tot.	72.43	92.39	89.78	66.03	205.93	147.25	146.15	220.69	185.93

Tableau A.7: Erreur maximale pour TD8 dans le PTVCDS

n \ s	1			(n-1)/2			(n-1)		
f	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
6	*	*	*	*	*	*	*	*	*
7	*	*	*	*	*	*	*	*	*
8	*	*	*	*	*	*	*	*	*
9	*	*	*	*	*	*	*	*	*
10	*	*	*	*	*	*	*	*	*
11	*	*	*	*	*	*	*	*	0.09
12	*	*	*	0.93	*	*	*	*	0.09
13	*	*	*	*	*	*	*	*	*
14	*	*	*	*	*	*	0.11	0.62	*
15	*	*	*	*	*	*	*	*	0.09
16	0.07	*	*	0.94	*	0.24	0.33	0.10	0.11
18	*	0.65	*	*	*	*	0.02	*	*
21	3.13	0.24	*	0.59	0.62	*	0.40	0.52	0.76
22	3.19	2.97	*	0.29	0.01	0.13	0.27	0.79	0.09
28	1.18	1.79	1.29	0.33	1.15	0.09	0.34	0.09	0.38
34	2.46	1.04	0.69	0.99	0.71	*	0.99	*	0.73
40	5.68	2.00	1.73	1.25	1.14	1.43	0.21	0.38	0.16
46	5.15	4.64	1.94	7.85	0.43	1.31	0.27	0.47	0.66
52	3.16	3.82	2.06	1.07	2.57	3.36	0.54	1.57	4.52
Er. max. globale	5.68	4.64	2.06	7.85	2.57	3.36	0.99	1.57	4.52
Er. max. globale par s		*1*			*(n-1)/2*			*(n-1)*	
		5.68			7.85			4.52	
Er. max. globale par f		*0.25*			*0.5*			*0.75*	
		7.85			4.64			4.52	
Temps moy. tot.	138.73	210.93	156.90	113.13	336.52	235.07	242.36	355.60	309.36

Tableau A.8: Erreur maximale pour T16 dans le PTVCDS

n \ s	1			(n-1)/2			(n-1)		
f	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
6	*	*	*	*	*	*	*	*	*
7	*	*	*	*	*	*	*	*	*
8	*	*	*	*	*	*	*	*	*
9	*	*	*	*	*	*	*	*	*
10	*	*	*	*	*	*	*	*	*
11	*	*	*	*	*	*	*	*	*
12	*	*	*	*	*	*	0.30	*	*
13	*	*	*	*	*	*	0.21	*	*
14	*	0.27	*	*	*	*	*	0.62	*
15	*	*	*	*	*	0.04	*	*	*
16	0.79	4.31	*	0.94	0.39	0.26	0.12	0.10	0.11
18	*	0.13	0.11	*	*	*	0.02	*	0.16
21	3.74	0.24	*	0.07	0.40	*	0.80	0.17	0.34
22	1.62	3.96	1.10	0.29	0.03	0.05	0.18	0.79	0.05
28	2.18	1.79	1.80	0.31	1.97	0.74	0.56	0.74	0.60
34	3.04	3.03	1.92	2.37	1.53	0.53	2.37	0.53	1.52
40	3.18	2.21	4.57	5.39	1.14	2.02	0.73	0.38	0.29
46	6.05	9.41	3.75	9.35	1.80	3.09	1.64	1.41	4.18
52	4.28	6.48	4.02	2.84	1.89	3.69	1.99	2.48	2.39
Er. max. globale	6.05	9.41	4.57	9.35	1.97	3.69	2.37	2.48	4.18
Er. max. globale par s		*1*			*(n-1)/2*			*(n-1)*	
		9.41			9.35			4.18	
Er. max. globale par f		*0.25*			*0.5*			*0.75*	
		9.35			9.41			4.57	
Temps moy. tot.	150.52	170.19	184.99	137.17	438.47	306.07	324.12	398.66	373.09

Tableau A.9: Erreur moyenne et nombre d'erreurs pour TD1 dans le PTVCDS

n \ s / f	1			(n-1)/2			(n-1)		
	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
6	*	*	*	*	*	*	*	*	*
7	*	*	*	*	*	*	0.10[1]	*	*
8	*	*	*	*	*	*	*	*	*
9	0.23[1]	*	*	*	*	*	*	*	*
10	*	*	*	0.13[1]	0.04[1]	*	*	0.11[1]	*
11	0.03[1]	*	*	*	*	*	*	*	*
12	0.25[1]	*	*	0.12[1]	*	*	0.06[1]	0.10[4]	0.02[3]
13	0.29[4]	0.10[3]	0.03[2]	0.02[2]	*	*	0.02[2]	0.02[1]	*
14	0.11[3]	0.07[3]	0.06[1]	0.34[3]	0.04[1]	*	0.26[2]	0.17[3]	0.03[1]
15	*	0.02[1]	0.09[1]	0.18[2]	0.02[1]	*	0.10[3]	0.10[4]	0.02[3]
16	0.08[1]	0.04[2]	*	0.08[1]	0.13[3]	0.03[2]	0.37[8]	0.11[5]	0.09[2]
18	0.21[2]	0.25[3]	0.04[1]	0.02[1]	0.07[4]	0.20[1]	0.02[3]	0.04[2]	0.05[2]
21	0.44[3]	0.76[3]	0.03[1]	0.09[5]	0.13[1]	*	0.04[4]	0.09[6]	0.80[6]
22	1.02[7]	0.36[2]	0.13[1]	1.93[7]	0.16[6]	0.27[1]	0.17[7]	0.13[6]	1.44[4]
28	1.42[8]	1.11[6]	0.60[4]	0.57[6]	0.30[6]	0.25[5]	0.55[6]	0.25[5]	0.04[3]
34	1.15[9]	1.35[7]	1.19[6]	0.58[9]	0.70[6]	0.05[4]	0.58[9]	0.05[4]	0.49[8]
40	2.79[9]	1.48[7]	1.49[7]	1.15[9]	1.03[10]	0.83[8]	0.50[6]	0.50[7]	0.45[6]
46	3.01[8]	2.92[6]	1.63[8]	2.47[4]	1.54[6]	0.76[5]	2.23[9]	0.40[9]	0.61[7]
52	2.40[9]	2.31[6]	0.83[4]	2.67[9]	1.38[8]	1.37[5]	3.47[10]	1.40[8]	1.32[7]
Er. moy. tot.	0.71[67]	0.57[49]	0.32[36]	0.54[60]	0.29[56]	0.20[33]	0.45[71]	0.18[65]	0.28[54]
Er. moy. tot. par s	*1* 0.53[152]			*(n-1)/2* 0.34[149]			*(n-1)* 0.30[190]		
Er. moy. tot. par f	*0.25* 0.57[198]			*0.5* 0.35[170]			*0.75* 0.27[123]		
Temps moy. tot.	18.26	26.71	18.54	15.05	42.05	30.37	29.66	46.85	38.91

Tableau A.10: Erreur moyenne et nombre d'erreurs pour T2 dans le PTVCDS

n \ s / f	1			(n-1)/2			(n-1)		
	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
6	*	*	*	*	*	*	*	*	*
7	*	*	*	*	*	*	*	*	*
8	*	*	*	*	*	*	*	*	*
9	*	*	*	*	*	*	*	*	0.01[1]
10	*	*	*	0.20[1]	*	0.01[1]	*	0.11[1]	*
11	*	0.13[3]	*	0.07[1]	*	*	0.04[1]	*	*
12	0.25[1]	0.25[2]	*	0.12[1]	*	*	0.11[1]	0.09[4]	0.04[3]
13	0.32[6]	0.10[3]	0.03[2]	0.16[3]	*	*	0.03[3]	*	*
14	0.41[5]	0.13[3]	0.04[1]	0.44[2]	0.04[1]	0.05[1]	0.20[2]	0.19[2]	*
15	0.01[2]	0.03[2]	0.13[2]	0.34[2]	*	0.01[2]	0.10[3]	0.09[4]	0.04[3]
16	0.29[3]	0.55[4]	*	0.34[2]	0.23[4]	0.04[2]	0.31[8]	0.16[6]	0.07[2]
18	1.16[6]	0.26[6]	0.01[1]	0.02[1]	0.08[5]	*	0.10[4]	0.06[4]	0.20[3]
21	0.80[7]	1.06[5]	*	0.28[8]	0.11[3]	*	0.44[7]	0.30[8]	0.66[8]
22	1.35[7]	0.86[5]	0.22[3]	2.18[7]	0.17[5]	0.27[2]	0.44[7]	1.57[7]	0.45[5]
28	2.16[10]	1.64[8]	0.98[7]	0.82[6]	0.58[7]	0.14[3]	0.81[6]	0.14[3]	0.29[6]
34	1.33[9]	1.21[6]	1.41[7]	1.58[8]	0.64[7]	0.50[6]	1.58[8]	0.50[6]	0.69[10]
40	4.65[9]	0.93[7]	2.04[8]	1.42[10]	1.87[9]	1.65[8]	0.43[6]	0.63[8]	0.41[7]
46	5.11[8]	3.04[6]	3.29[8]	2.81[7]	2.38[8]	3.72[8]	1.80[10]	1.56[10]	0.88[8]
52	4.56[9]	4.83[8]	2.73[7]	2.38[10]	2.48[10]	1.48[7]	1.81[10]	1.13[8]	2.06[9]
Er. moy. tot.	1.18[82]	0.79[68]	0.57[46]	0.69[69]	0.45[61]	0.42[42]	0.43[77]	0.34[71]	0.31[67]
Er. moy. tot. par s	*1* 0.85[196]			*(n-1)/2* 0.52[172]			*(n-1)* 0.36[215]		
Er. moy. tot. par f	*0.25* 0.77[228]			*0.5* 0.53[200]			*0.75* 0.43[155]		
Temps moy. tot.	18.63	22.08	22.60	17.08	51.09	40.33	40.21	57.75	49.90

Tableau A.11: Erreur moyenne et nombre d'erreurs pour TD2 dans le PTVCDS

n \ s	1			(n-1)/2			(n-1)		
f	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
6	*	*	*	*	*	*	*	*	*
7	*	*	*	*	*	*	*	*	*
8	*	*	*	*	*	*	*	*	*
9	*	*	*	*	*	*	*	*	0.01[1]
10	*	*	*	0.13[1]	*	0.01[1]	*	0.11[1]	*
11	*	0.07[1]	*	0.07[1]	*	*	*	*	*
12	0.04[1]	*	*	*	*	*	0.05[1]	0.08[3]	0.02[3]
13	0.06[3]	0.05[1]	0.03[2]	*	*	*	0.02[1]	*	*
14	0.01[1]	0.03[1]	0.04[1]	0.01[1]	0.04[1]	0.02[1]	0.20[1]	*	*
15	*	0.02[1]	*	0.12[1]	*	*	0.02[1]	0.08[3]	0.02[3]
16	0.09[2]	0.30[3]	*	0.08[1]	0.09[3]	0.03[1]	0.24[6]	0.11[4]	*
18	0.25[3]	0.19[4]	0.01[1]	0.02[1]	*	*	0.01[3]	0.02[1]	0.16[2]
21	0.37[6]	0.47[3]	*	0.11[4]	0.11[3]	*	0.07[3]	0.14[6]	0.20[5]
22	0.64[4]	0.40[2]	0.22[2]	1.88[5]	0.08[3]	0.27[2]	0.28[4]	0.12[4]	0.44[4]
28	1.69[8]	0.78[5]	0.34[2]	0.44[5]	0.26[5]	0.10[3]	0.43[5]	0.10[3]	0.15[3]
34	0.64[8]	0.19[3]	0.96[6]	0.81[8]	0.21[6]	0.22[5]	0.81[8]	0.22[5]	0.49[9]
40	1.49[7]	0.67[6]	1.04[5]	0.92[9]	0.67[7]	0.54[5]	0.35[6]	0.44[7]	0.25[5]
46	1.56[8]	1.63[6]	0.80[5]	0.53[6]	0.42[4]	0.98[7]	0.33[8]	0.60[10]	0.53[8]
52	2.30[9]	2.29[8]	1.08[7]	1.49[10]	1.57[8]	0.43[6]	0.59[7]	0.72[7]	1.53[9]
Er. moy. tot.	0.48[61]	0.37[44]	0.24[31]	0.35[53]	0.18[43]	0.14[32]	0.18[54]	0.14[54]	0.20[54]
Er. moy. tot. par s	*1*			*(n-1)/2*			*(n-1)*		
	0.36[136]			0.22[128]			0.17[162]		
Er. moy. tot. par f	*0.25*			*0.5*			*0.75*		
	0.34[168]			0.23[141]			0.19[117]		
Temps moy. tot.	35.66	53.66	37.84	29.55	87.37	63.61	67.26	93.81	81.33

Tableau A.12: Erreur moyenne et nombre d'erreurs pour T4 dans le PTVCDS

n \ s	1			(n-1)/2			(n-1)		
f	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
6	*	*	*	*	*	*	*	*	*
7	1.13[1]	*	*	*	*	*	*	*	*
8	*	*	*	*	*	*	*	*	*
9	*	*	*	*	*	*	*	*	*
10	*	*	*	*	*	*	*	*	*
11	*	*	*	*	*	*	*	*	*
12	0.04[1]	*	*	0.12[1]	*	*	0.10[1]	0.08[3]	0.04[3]
13	0.08[3]	*	*	*	*	*	0.02[2]	*	*
14	0.11[2]	0.02[2]	*	0.04[1]	*	0.05[1]	0.20[2]	0.09[1]	*
15	*	0.21[1]	0.13[2]	0.13[1]	*	*	0.08[2]	0.08[3]	0.04[3]
16	0.13[3]	0.22[2]	*	0.09[1]	0.09[2]	0.04[2]	0.19[6]	0.04[5]	0.01[2]
18	0.62[4]	0.18[6]	0.01[1]	0.02[1]	*	*	*	*	0.18[2]
21	0.80[8]	0.11[3]	*	0.26[7]	0.11[3]	*	0.33[5]	0.09[7]	0.30[6]
22	0.79[5]	0.81[4]	0.22[2]	0.39[7]	0.12[4]	0.02[2]	0.15[8]	0.12[5]	*
28	1.64[8]	1.47[7]	0.98[7]	1.04[6]	0.30[6]	0.19[4]	1.03[6]	0.19[4]	0.23[5]
34	1.24[9]	1.20[6]	1.44[7]	0.57[6]	0.38[7]	0.33[6]	0.57[6]	0.33[6]	0.60[10]
40	3.83[9]	0.86[6]	1.64[7]	1.11[10]	1.08[9]	0.82[7]	0.47[6]	0.51[8]	0.29[6]
46	3.38[8]	2.13[6]	2.53[8]	1.48[7]	1.12[7]	2.48[8]	0.86[9]	1.02[10]	1.53[9]
52	3.46[9]	3.27[8]	1.73[8]	1.40[9]	1.24[8]	1.52[8]	1.65[10]	0.92[8]	1.84[9]
Er. moy. tot.		0.64[164]			0.29[146]			0.25[182]	
0.91[70]	0.55[52]	0.46[42]	0.35[57]	0.23[49]	0.29[40]	0.30[63]	0.18[61]	0.27[58]	
Er. moy. tot. par s		*1*			*(n-1)/2*			*(n-1)*	
		0.64[164]			0.29[146]			0.25[182]	
Er. moy. tot. par f		*0.25*			*0.5*			*0.75*	
		0.52[190]			0.32[162]			0.34[140]	
Temps moy. tot.	40.56	43.23	46.14	34.61	106.62	80.69	79.92	103.98	87.26

Tableau A.13: Erreur moyenne et nombre d'erreurs pour TD4 dans le PTVCDS

n \ s \ f	1			(n-1)/2			(n-1)		
	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
6	*	*	*	*	*	*	*	*	*
7	*	*	*	*	*	*	*	*	*
8	*	*	*	*	*	*	*	*	*
9	*	*	*	*	*	*	*	*	*
10	*	*	*	*	*	*	*	*	*
11	*	*	*	*	*	*	*	*	*
12	0.04[1]	*	*	0.12[1]	*	*	*	0.06[3]	*
13	*	*	*	*	*	*	0.02[1]	*	*
14	*	*	*	*	*	0.05[1]	0.20[2]	0.09[1]	*
15	*	0.02[1]	*	*	*	*	*	0.06[3]	*
16	0.08[1]	0.01[1]	*	0.09[1]	0.06[1]	0.03[2]	0.03[2]	0.03[4]	0.01[2]
18	0.18[2]	0.07[2]	*	0.02[1]	*	*	*	*	*
21	0.18[5]	0.07[2]	*	0.20[4]	0.11[3]	*	0.03[2]	0.03[4]	0.04[4]
22	0.32[1]	0.40[1]	0.09[1]	0.14[4]	0.05[3]	0.01[2]	0.05[5]	0.11[4]	*
28	1.10[8]	0.27[4]	0.17[2]	0.45[6]	0.21[4]	0.11[3]	0.43[5]	0.11[3]	0.15[3]
34	0.64[9]	0.35[5]	0.42[4]	0.39[5]	0.15[3]	0.15[3]	0.39[5]	0.15[3]	0.13[5]
40	0.87[6]	0.79[5]	0.27[2]	0.60[8]	0.25[6]	0.47[4]	0.23[4]	0.14[6]	*
46	1.57[8]	1.07[6]	1.15[8]	0.17[3]	0.28[4]	0.55[5]	0.10[7]	0.21[8]	0.30[7]
52	1.51[9]	1.50[7]	0.95[7]	0.77[8]	0.97[6]	0.56[5]	0.59[9]	0.68[7]	0.44[7]
Er. moy. tot.	0.34[50]	0.24[35]	0.16[24]	0.16[41]	0.11[30]	0.10[27]	0.11[43]	0.09[46]	0.06[37]
Er. moy. tot. par s	*1*			*(n-1)/2*			*(n-1)*		
	0.25[109]			0.12[98]			0.09[126]		
Er. moy. tot. par f	*0.25*			*0.5*			*0.75*		
	0.20[134]			0.15[111]			0.11[88]		
Temps moy. tot.	74.71	104.15	77.37	59.31	174.03	125.88	129.35	173.30	151.16

Tableau A.14: Erreur moyenne et nombre d'erreurs pour T8 dans le PTVCDS

n \ s / f	1			(n-1)/2			(n-1)		
	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
6	*	*	*	*	*	*	*	*	*
7	*	*	*	*	*	*	*	*	*
8	*	*	*	*	*	*	*	*	*
9	*	*	*	*	*	*	*	*	*
10	*	*	*	*	*	*	*	*	*
11	*	*	*	*	*	*	*	*	*
12	0.03[1]	*	*	0.09[1]	*	*	0.03[1]	*	0.04[2]
13	0.06[2]	*	*	*	*	*	0.02[1]	*	*
14	0.01[1]	0.03[2]	*	0.04[1]	*	*	0.06[2]	0.06[1]	*
15	*	0.19[1]	*	*	*	*	*	*	0.04[2]
16	0.10[2]	0.43[1]	*	0.09[1]	*	0.03[1]	0.17[3]	0.03[4]	0.01[2]
18	0.46[2]	0.09[3]	*	*	*	*	*	0.01[1]	0.02[2]
21	0.61[6]	0.03[2]	*	0.18[5]	0.10[3]	*	0.19[5]	0.09[6]	0.13[3]
22	0.60[4]	0.43[4]	0.22[2]	0.12[5]	0.10[3]	0.02[2]	0.08[5]	0.11[4]	0.01[3]
28	0.71[7]	0.97[6]	0.23[3]	0.30[5]	0.28[5]	0.22[4]	0.27[5]	0.22[4]	0.12[4]
34	1.03[8]	1.02[6]	0.94[7]	0.52[6]	0.53[9]	0.03[3]	0.52[6]	0.03[3]	0.61[10]
40	3.32[9]	0.86[6]	1.55[7]	1.62[10]	0.64[8]	0.55[8]	0.35[7]	0.10[8]	0.18[7]
46	2.84[8]	2.40[6]	2.33[8]	1.97[7]	0.33[6]	1.15[8]	0.48[9]	1.16[10]	0.44[7]
52	3.58[9]	2.49[8]	1.81[9]	1.28[8]	0.79[7]	1.50[8]	0.51[10]	0.66[8]	1.10[8]
Er. moy. tot.	0.70[59]	0.47[45]	0.37[36]	0.33[49]	0.15[42]	0.18[35]	0.14[56]	0.13[49]	0.14[51]
Er. moy. tot. par s	*1* 0.51[140]			*(n-1)/2* 0.22[126]			*(n-1)* 0.14[156]		
Er. moy. tot. par f	*0.25* 0.39[164]			*0.5* 0.25[136]			*0.75* 0.23[122]		
Temps moy. tot.	72.43	92.39	89.78	66.03	205.93	147.25	146.15	220.69	185.93

Tableau A.15: Erreur moyenne et nombre d'erreurs pour TD8 dans le PTVCDS

n \ s / f	1			(n-1)/2			(n-1)		
	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
6	*	*	*	*	*	*	*	*	*
7	*	*	*	*	*	*	*	*	*
8	*	*	*	*	*	*	*	*	*
9	*	*	*	*	*	*	*	*	*
10	*	*	*	*	*	*	*	*	*
11	*	*	*	*	*	*	*	*	*
12	*	*	*	0.09[1]	*	*	*	*	0.02[2]
13	*	*	*	*	*	*	*	*	*
14	*	*	*	*	*	*	0.01[1]	0.06[1]	*
15	*	*	*	*	*	*	*	*	0.02[2]
16	*	*	*	0.09[1]	*	0.02[1]	0.05[2]	0.01[2]	0.01[1]
18	*	0.07[1]	*	*	*	*	*	*	*
21	0.36[4]	0.02[1]	*	0.06[1]	0.10[3]	*	0.07[4]	0.08[5]	0.11[3]
22	0.32[1]	0.32[2]	*	0.03[1]	*	0.02[2]	0.04[2]	0.09[3]	0.01[3]
28	0.39[7]	0.53[5]	0.13[1]	0.05[3]	0.17[3]	*	0.04[3]	*	0.07[2]
34	0.49[7]	0.18[3]	0.25[4]	0.12[2]	0.10[4]	*	0.12[2]	*	0.17[6]
40	1.12[6]	0.68[4]	0.20[2]	0.31[9]	0.23[5]	0.23[4]	0.07[5]	0.10[7]	0.03[4]
46	1.68[6]	1.01[5]	0.52[6]	1.09[6]	0.08[5]	0.45[7]	0.08[7]	0.14[6]	0.17[5]
52	1.65[9]	1.49[8]	0.90[7]	0.28[6]	0.45[3]	0.59[5]	0.13[6]	0.24[4]	0.72[3]
Er. moy. tot.	0.32[41]	0.23[29]	0.10[20]	0.11[30]	0.06[24]	0.07[21]	0.03[33]	0.04[30]	0.07[32]
Er. moy. tot. par s	*1*			*(n-1)/2*			*(n-1)*		
	0.22[90]			0.08[75]			0.05[95]		
Er. moy. tot. par f	*0.25*			*0.5*			*0.75*		
	0.15[104]			0.11[83]			0.08[73]		
Temps moy. tot.	138.73	210.93	156.90	113.13	336.52	235.07	242.36	355.60	369.36

Tableau A.16: Erreur moyenne et nombre d'erreurs pour T16 dans le PTVCDS

n \ s \ f	1			(n-1)/2			(n-1)		
	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
6	*	*	*	*	*	*	*	*	*
7	*	*	*	*	*	*	*	*	*
8	*	*	*	*	*	*	*	*	*
9	*	*	*	*	*	*	*	*	*
10	*	*	*	*	*	*	*	*	*
11	*	*	*	*	*	*	*	*	*
12	*	*	*	*	*	*	0.03[1]	*	*
13	*	*	*	*	*	*	0.02[1]	*	*
14	*	0.03[2]	*	*	*	*	*	0.06[1]	*
15	*	*	*	*	*	*	*	*	*
16	0.08[1]	0.43[1]	*	0.09[1]	0.04[1]	0.03[1]	0.01[1]	0.02[3]	0.01[1]
18	*	0.01[1]	0.01[1]	*	*	*	*	*	0.02[1]
21	0.47[3]	0.03[2]	*	0.01[2]	0.05[2]	*	0.08[2]	0.02[4]	0.05[4]
22	0.36[4]	0.41[2]	0.20[2]	0.04[3]	*	*	0.02[2]	0.12[5]	*
28	0.73[7]	0.47[5]	0.20[2]	0.04[3]	0.28[5]	0.24[5]	0.09[3]	0.24[5]	0.12[5]
34	1.39[9]	0.82[5]	0.39[3]	0.52[6]	0.43[8]	0.07[4]	0.52[6]	0.07[4]	0.51[10]
40	1.57[9]	1.07[7]	1.04[7]	1.13[7]	0.36[7]	0.34[6]	0.17[6]	0.10[8]	0.06[8]
46	2.49[8]	2.04[6]	1.42[6]	1.18[5]	0.26[4]	0.95[7]	0.38[9]	0.29[8]	0.48[6]
52	2.15[9]	1.97[8]	1.73[9]	0.85[9]	0.68[9]	1.30[8]	0.42[10]	0.67[8]	0.77[8]
Er. moy. tot.	0.49[50]	0.38[39]	0.26[30]	0.20[36]	0.11[37]	0.15[33]	0.09[42]	0.08[46]	0.11[44]
Er. moy. tot. par s	*1*			*(n-1)/2*			*(n-1)*		
	0.38[119]			0.16[106]			0.09[132]		
Er. moy. tot. par f	*0.25*			*0.5*			*0.75*		
	0.26[128]			0.1[122]			0.17[107]		
Temps moy. tot.	150.52	170.19	184.99	137.17	438.47	306.07	324.12	398.66	373.09

Annexe B

Erreurs maximales et erreurs moyennes par taille dans le PVCCS

Les tableaux de cet annexe se suivent par paire (TDi, T2i), $i = 4, 8$ pour chaque mesure de performance (erreur maximale et erreur moyenne).

Les tableaux B.2 et B.4 présentent les erreurs maximales *absolues* par taille, de l'algorithme tabou (AT) simple pour les 5 familles de problèmes définies dans le PVCCS, selon s, pour les versions T8 et T16 respectivement. Les tableaux B.1 et B.3, présentent les erreurs maximales *absolues* par taille, de l'algorithme tabou avec diversification (ATD) pour les mêmes familles de problèmes pour les versions TD4 et TD8 respectivement.

De même, les tableaux B.6 et B.8 présentent les erreurs moyennes et le nombre d'erreurs par taille pour T8 et T16 respectivement et, les tableaux B.5 et B.7 présentent ces mesure de performance pour TD4 et TD8 respectivement.

Tableau B.1: Erreur maximale pour TD4 dans le PVCCS

	Intervalles des probablilités de présence des clients				
$[a\ b]$	[0.2 0.2]	[0.5 0.5]	[0.8 0.8]	[0.1 1]	[0.5 1]
$n_{[s]}$					
$10_{[5]}$	*	*	*	*	*
$10_{[3]}$	*	*	*	*	*
$20_{[10]}$	*	*	0.76	*	1.79
$20_{[6]}$	0.62	1.59	5.50	0.22	*
$30_{[9]}$	2.06	5.82	5.84	5.80	5.80
$30_{[6]}$	*	4.97	5.74	3.89	5.71
$40_{[8]}$	1.97	4.06	1.03	6.58	3.94
$40_{[4]}$	4.18	4.88	7.90	7.41	3.09
$50_{[5]}$	9.53	1.45	6.98	9.54	7.58
Erreur. max. globale	9.53	5.82	7.90	9.54	7.58
Temps. moy. total	44.65	41.57	41.89	43.50	42.72

Tableau B.2: Erreur maximale pour T8 dans le PVCCS

	Intervalles des probablilités de présence des clients				
$[a\ b]$	[0.2 0.2]	[0.5 0.5]	[0.8 0.8]	[0.1 1]	[0.5 1]
$n_{[s]}$					
$10_{[5]}$	*	*	*	*	*
$10_{[3]}$	*	*	*	*	*
$20_{[10]}$	*	4.34	2.32	3.15	1.97
$20_{[6]}$	6.43	2.52	5.50	0.28	4.49
$30_{[9]}$	5.63	7.70	6.00	5.48	7.48
$30_{[6]}$	7.98	6.97	6.65	5.00	7.59
$40_{[8]}$	8.87	11.50	11.04	10.18	15.45
$40_{[4]}$	6.68	13.37	8.12	9.68	5.24
$50_{[5]}$	17.01	7.84	17.45	15.59	15.70
Erreur. max. globale	17.01	13.37	17.45	15.59	15.70
Temps. moy. total	37.88	30.60	29.76	31.86	31.42

Tableau B.3: Erreur maximale pour TD8 dans le PVCCS

	Intervalles des probablilités de présence des clients				
$[a\ b]$	[0.2 0.2]	[0.5 0.5]	[0.8 0.8]	[0.1 1]	[0.5 1]
$n_{[s]}$					
$10_{[5]}$	*	*	*	*	*
$10_{[3]}$	*	*	*	*	*
$20_{[10]}$	*	0.05	*	*	0.22
$20_{[6]}$	0.79	0.08	0.17	*	4.49
$30_{[9]}$	0.59	1.12	1.42	2.75	3.95
$30_{[6]}$	2.45	3.94	5.22	3.89	4.87
$40_{[8]}$	2.49	3.74	2.00	3.00	4.92
$40_{[4]}$	3.54	1.13	4.72	5.16	4.62
$50_{[5]}$	5.34	5.62	7.45	7.85	7.58
Erreur. max. globale	5.34	5.62	7.45	7.85	7.58
Temps. moy. total	92.31	80.51	79.50	81.91	81.67

Tableau B.4: Erreur maximale pour T16 dans le PVCCS

	Intervalles des probablilités de présence des clients				
$[a\ b]$	[0.2 0.2]	[0.5 0.5]	[0.8 0.8]	[0.1 1]	[0.5 1]
$n_{[s]}$					
$10_{[5]}$	*	*	*	*	*
$10_{[3]}$	*	*	*	*	*
$20_{[10]}$	*	4.34	2.32	3.15	1.97
$20_{[6]}$	3.11	0.19	5.50	0.34	5.26
$30_{[9]}$	3.02	7.21	4.85	5.48	7.48
$30_{[6]}$	9.36	3.84	6.71	5.46	5.71
$40_{[8]}$	5.56	11.17	15.98	8.04	9.57
$40_{[4]}$	8.55	7.97	7.96	7.46	12.39
$50_{[5]}$	15.12	6.52	16.75	17.69	15.01
Erreur. max. globale	15.12	11.17	16.75	17.69	15.01
Temps. moy. total	84.86	77.80	113.03	75.85	70.59

Tableau B.5: Erreur moyenne[*#d'erreurs*] pour TD4 dans le PVCCS

	Intervalles des probablilités de présence des clients				
$[a\ b]$	[0.2 0.2]	[0.5 0.5]	[0.8 0.8]	[0.1 1]	[0.5 1]
$n_{[s]}$					
10[5]	*	*	*	*	*
10[3]	*	*	*	*	*
20[10]	*	*	0.08[1]	*	0.18[1]
20[6]	0.09[2]	0.16[1]	0.57[3]	0.02[1]	*
30[9]	0.21[2]	0.78[3]	1.07[3]	1.08[3]	1.14[4]
30[6]	*	0.93[3]	2.47[7]	0.76[4]	1.19[5]
40[8]	0.26[4]	0.76[5]	0.24[4]	1.41[4]	0.50[3]
40[4]	0.71[3]	0.97[5]	2.09[8]	1.88[8]	0.86[5]
50[5]	3.00[8]	0.30[3]	1.91[6]	1.78[5]	1.98[4]
Erreur. moy. totale	0.47[19]	0.43[20]	0.94[32]	0.77[25]	0.65[22]
Temps. moy. total	44.65	41.57	41.89	43.50	42.72

Tableau B.6: Erreur moyenne[*#d'erreurs*] pour T8 dans le PVCCS

	Intervalles des probablilités de présence des clients				
$[a\ b]$	[0.2 0.2]	[0.5 0.5]	[0.8 0.8]	[0.1 1]	[0.5 1]
$n_{[s]}$					
10[5]	*	*	*	*	*
10[3]	*	*	*	*	*
20[10]	*	0.44[2]	0.29[2]	0.31[1]	0.22[2]
20[6]	1.12[5]	0.28[4]	1.22[4]	0.07[5]	0.87[5]
30[9]	1.35[6]	2.23[6]	2.35[8]	1.10[3]	2.23[6]
30[6]	3.01[7]	1.69[3]	2.79[9]	2.24[8]	3.08[8]
40[8]	3.54[8]	6.00[8]	4.55[8]	4.29[8]	4.16[8]
40[4]	2.69[9]	3.91[8]	4.86[10]	4.85[9]	2.88[9]
50[5]	8.86[10]	3.15[9]	8.31[9]	5.76[10]	7.60[9]
Erreur. moy. totale	2.29[45]	1.97[40]	2.71[50]	2.07[44]	2.34[47]
Temps. moy. total	37.88	30.60	29.76	31.86	31.42

Tableau B.7: Erreur moyenne[*#d'erreurs*] pour TD8 dans le PVCCS

	Intervalles des probablilités de présence des clients				
$[a\ b]$	[0.2 0.2]	[0.5 0.5]	[0.8 0.8]	[0.1 1]	[0.5 1]
$n_{[s]}$					
10[5]	*	*	*	*	*
10[3]	*	*	*	*	*
20[10]	*	*	*	*	0.02[1]
20[6]	0.14[2]	*	0.02[2]	*	0.45[2]
30[9]	0.06[1]	0.12[2]	0.17[2]	0.44[2]	0.43[2]
30[6]	0.36[2]	0.39[1]	1.76[7]	0.64[3]	0.89[5]
40[8]	0.49[3]	0.65[5]	0.56[4]	0.56[3]	1.57[4]
40[4]	0.76[6]	0.20[2]	1.45[6]	1.05[7]	1.04[5]
50[5]	1.56[5]	0.98[5]	1.90[8]	1.67[6]	3.29[7]
Erreur. moy. totale	0.37[19]	0.26[15]	0.65[29]	0.48[21]	0.86[26]
Temps. moy. total	92.31	80.51	79.50	81.91	81.67

Tableau B.8: Erreur moyenne[*#d'erreurs*] pour T16 dans le PVCCS

	Intervalles des probablilités de présence des clients				
$[a\ b]$	[0.2 0.2]	[0.5 0.5]	[0.8 0.8]	[0.1 1]	[0.5 1]
$n_{[s]}$					
10[5]	*	*	*	*	*
10[3]	*	*	*	*	*
20[10]	*	0.44[2]	0.29[2]	0.31[1]	0.22[2]
20[6]	0.41[3]	0.03[2]	0.64[3]	0.11[5]	0.93[5]
30[9]	0.64[4]	2.00[6]	1.85[7]	1.05[3]	1.65[5]
30[6]	2.13[6]	0.43[2]	2.75[10]	1.94[7]	2.55[8]
40[8]	2.07[8]	4.88[8]	4.72[8]	3.80[8]	3.43[8]
40[4]	2.10[7]	3.14[8]	3.60[10]	3.85[9]	3.83[9]
50[5]	6.84[10]	2.84[8]	6.58[8]	5.37[10]	7.24[9]
Erreur. moy. totale	1.58[38]	1.53[36]	2.27[48]	1.83[43]	2.21[46]
Temps. moy. total	84.86	77.80	113.03	75.85	70.59

Tableau B.9: Erreur maximale de l'algorithme tabou+diversification selon le type de problèmes dans le PVCCS

	Intervalles des probablilités de présence des clients				
$[a\ b]$	[0.2 0.2]	[0.5 0.5]	[0.8 0.8]	[0.1 1]	[0.5 1]
Erreur Maximale					
s^+ TD4	9.53	5.82	6.98	9.54	7.58
s^-	4.18	4.97	7.90	7.41	5.71
s^+ TD8	5.34	5.62	7.45	7.85	7.58
s^-	3.54	3.94	5.22	5.16	4.87

Tableau B.10: Erreur maximale de l'algorithme tabou selon le type de problèmes dans le PVCCS

	Intervalles des probablilités de présence des clients				
$[a\ b]$	[0.2 0.2]	[0.5 0.5]	[0.8 0.8]	[0.1 1]	[0.5 1]
Erreur Maximale					
s^+ T8	17.01	11.50	17.45	15.59	15.70
s^-	7.98	13.37	8.12	9.68	7.59
s^+ T16	15.12	11.17	16.75	17.69	15.01
s^-	9.36	7.97	7.96	7.46	12.39

Tableau B.11: Erreur moyenne de l'algorithme tabou+diversification selon le type de problèmes dans le PVCCS

	Intervalles des probablilités de présence des clients				
$[a\ b]$	[0.2 0.2]	[0.5 0.5]	[0.8 0.8]	[0.1 1]	[0.5 1]
Erreur Moyenne					
s^+ TD4	0.70[14]	0.37[11]	0.66[14]	0.85[12]	0.76[12]
s^-	0.20[5]	0.52[9]	1.28[18]	0.67[13]	0.51[10]
s^+ TD8	0.42[9]	0.35[13]	0.53[14]	0.53[11]	1.06[14]
s^-	0.32[10]	0.15[4]	0.81[15]	0.42[10]	0.60[12]

Tableau B.12: Erreur moyenne de l'algorithme tabou selon le type de problèmes dans le PVCCS

	Intervalles des probablilités de présence des clients				
$[a\ b]$	[0.2 0.2]	[0.5 0.5]	[0.8 0.8]	[0.1 1]	[0.5 1]
Erreur Moyenne					
s^+ T8	2.75[24]	2.36[25]	3.10[27]	2.29[22]	2.84[25]
s^-	1.71[21]	1.47[15]	2.22[23]	1.79[22]	1.71[22]
s^+ T16	1.91[22]	2.03[24]	2.69[25]	2.11[22]	2.51[24]
s^-	1.16[16]	0.90[12]	1.75[23]	1.47[21]	1.83[22]

Annexe C

Erreurs maximales et erreurs moyennes par taille dans le PTVDS

Les tableaux C.2 et C.1 présentent les erreurs maximales *absolues* par taille, de l'algorithme tabou (AT) simple et de l'algorithme tabou avec diversification (AT-D) respectivement pour les 5 familles de problèmes définies selon f dans le PTVDS, pour TD4 et T8. Les tableaux C.4 et C.3 présentent les erreurs moyennes par taille, de l'AT simple et de l'ATD respectivement, pour TD4 et T8. Les tableaux B.6

Tableau C.1: Erreur maximale pour TD4 dans le PTVDS

n \ f	0.3	0.5	0.7	0.9
10	*	*	*	*
20	0.21	0.29	*	0.05
30	3.41	2.96	0.24	0.49
40	4.33	4.55	2.61	1.47
50	5.54	4.66	1.44	3.34
60	10.64	10.52	4.95	4.46
70	8.70	8.15	3.47	4.30
Erreur. max. globale	10.64	10.52	4.95	4.46
Temps. moy. total	351.27	512.44	483.17	594.39

et B.8 présentent les erreurs moyennes par taille, de l'AT simple pour les 5 familles de problèmes définies dans le PVCCS, pour les versions T8 et T16 respectivement. Les tableaux B.5 et B.7, présentent les erreurs moyennes par taille, de l'ATD pour

Tableau C.2: Erreur maximale pour T8 dans le PTVDS

n \ f	0.3	0.5	0.7	0.9
10	*	*	*	*
20	0.52	0.29	*	0.62
30	3.41	5.16	3.22	2.42
40	6.17	4.89	6.49	2.09
50	6.08	5.62	4.04	4.78
60	10.86	10.74	7.57	9.52
70	9.31	8.49	18.00	10.16
Erreur. max. globale	10.86	10.74	18.00	10.16
Temps. moy. total	360.51	423.23	655.22	886.98

Tableau C.3: Erreur moyenne[*#d'erreurs*] pour TD4 dans le PTVDS

n \ f	0.3	0.5	0.7	0.9
10	*	*	*	*
20	0.04[2]	0.06[3]	*	*
30	0.62[6]	0.54[5]	0.02[1]	0.05[1]
40	1.56[8]	1.43[8]	0.59[3]	0.23[2]
50	2.35[10]	2.42[10]	0.35[5]	1.06[5]
60	3.72[10]	3.78[10]	1.27[5]	1.08[5]
70	3.49[9]	2.78[8]	1.09[7]	1.69[7]
Erreur. moy. totale	1.68[45]	1.57[44]	0.48[21]	0.59[21]
Temps. moy. total	351.27	512.44	483.17	594.39

les mêmes familles de problèmes, pour les versions TD4 et TD8 respectivement.

Tableau C.4: Erreur moyenne[*#d'erreurs*] pour T8 dans le PTVDS

n \ f	0.3	0.5	0.7	0.9
10	*	*	*	*
20	0.07[2]	0.05[3]	*	0.11[3]
30	1.23[9]	1.10[7]	0.65[5]	0.53[5]
40	2.07[9]	2.24[10]	1.34[6]	0.74[7]
50	3.40[10]	2.13[10]	1.46[8]	2.03[8]
60	5.88[10]	5.91[10]	3.91[9]	3.52[7]
70	5.18[10]	4.23[9]	2.78[9]	3.27[10]
Erreur. moy. totale	2.55[50]	2.24[49]	1.45[37]	1.46[40]
Temps. moy. total	360.51	423.23	655.22	886.98

Annexe D

Résultats détaillés de la section 4.5: erreurs moyennes par taille

Les tableaux D.1, D.2, D.3 et D.4 présentent les erreurs moyennes et le nombre d'erreurs (entre crochets, en indice) par taille pour toutes les familles de problèmes définies à la section 4.5, pour les versions DIV1, DIV2, DIV3, DIV4 de l'algorithme tabou avec diversification, également caractérisées dans cette section. Le tableau D.5 présente la même mesure de performance pour la version TAB de l'algorithme tabou, telle que définie à la section 4.5.

Tableau D.1: Erreurs[#*erreurs*] de l'algorithme tabou+diversification relativement à la meilleure solution connue: DIV1

s	1			(n-1)/2			(n-1)		
(n-m) f	0.25	0.50	0.75	0.25	0.50	0.75	0.25	0.50	0.75
(52-6)	0.38[6]	0.80[7]	0.66[4]	1.51[9]	1.36[8]	0.24[4]	1.18[6]	0.29[3]	0.84[7]
(76-3)	0.87[5]	1.08[7]	1.30[6]	0.78[5]	1.39[8]	0.88[4]	1.57[3]	1.73[7]	0.78[7]
(100-4)	1.58[9]	1.87[7]	1.10[6]	0.93[5]	1.57[5]	0.91[7]	0.33[4]	0.63[6]	0.60[7]
(76-8)	2.53[8]	1.38[9]	0.84[6]	9.42[10]	0.75[6]	1.17[7]	1.20[7]	1.22[9]	1.31[10]
(100-10)	1.10[7]	0.58[5]	1.68[7]	0.70[6]	0.82[5]	1.00[7]	3.09[8]	0.77[7]	0.65[6]
Erreurs moyennes	1.29[35]	1.14[35]	1.12[29]	2.67[35]	1.18[32]	0.84[29]	1.47[28]	0.93[32]	0.84[37]
Erreurs moyennes par *s*	1.18 [99]			1.56 [96]			1.08 [97]		

Tableau D.2: Erreurs[#*erreurs*] de l'algorithme tabou+diversification relativement à la meilleure solution connue: DIV2

s	1			(n-1)/2			(n-1)		
(n-m) f	0.25	0.50	0.75	0.25	0.50	0.75	0.25	0.50	0.75
(52-6)	0.42 [8]	1.86 [9]	1.37 [5]	1.90 [7]	1.27 [6]	0.37 [5]	0.82 [7]	0.83 [8]	0.22 [4]
(76-3)	0.85 [4]	0.84 [4]	1.57 [7]	0.26 [2]	1.35 [5]	1.94 [8]	0.37 [4]	0.40 [5]	0.71 [6]
(100-4)	0.87 [5]	0.86 [5]	0.99 [4]	1.11 [6]	1.68 [5]	0.79 [5]	0.13 [3]	0.19 [6]	0.87 [7]
(76-8)	2.81 [9]	1.73 [9]	1.33 [9]	5.15 [8]	0.65 [5]	1.01 [5]	4.68 [10]	1.45 [8]	0.72 [5]
(100-10)	1.71 [10]	1.27 [7]	1.19 [7]	1.92 [7]	1.60 [8]	1.32 [6]	1.25 [4]	1.13 [9]	0.72 [10]
Erreurs moyennes	1.33 [36]	1.31 [34]	1.29 [32]	2.07 [30]	1.31 [29]	1.09 [29]	1.45 [28]	0.80 [36]	0.65 [32]
Erreurs moyennes par *s*	1.31 [102]			1.49 [88]			0.97 [96]		

Tableau D.3: Erreurs[*#erreurs*] de l'algorithme tabou+diversification relativement à la meilleure solution connue: DIV3

s	1			(n-1)/2			(n-1)		
(n-m) f	0.25	0.50	0.75	0.25	0.50	0.75	0.25	0.50	0.75
(52-6)	0.53 [8]	0.91 [6]	0.75 [5]	1.88 [8]	1.24 [6]	0.10 [3]	1.15 [6]	0.82 [7]	0.42 [6]
(76-3)	0.89 [6]	1.11 [5]	0.53 [4]	2.46 [5]	2.91 [6]	1.97 [7]	3.79 [4]	3.42 [6]	1.02 [7]
(100-4)	1.50 [8]	2.27 [8]	1.77 [6]	1.16 [5]	1.33 [5]	1.25 [9]	0.33 [4]	0.18 [4]	1.75 [7]
(76-8)	2.03 [8]	0.37 [3]	1.19 [6]	3.22 [8]	0.64 [7]	1.07 [6]	1.98 [7]	0.69 [4]	0.91 [8]
(100-10)	2.33 [9]	0.91 [7]	1.04 [4]	4.21 [8]	1.18 [8]	1.28 [7]	5.40 [8]	1.79 [5]	0.85 [7]
Erreurs moyennes	1.46 [39]	1.11 [29]	1.06 [25]	2.59 [34]	1.46 [32]	1.14 [32]	2.53 [29]	1.38 [26]	0.99 [35]
Erreurs moyennes par *s*	1.21 [93]			1.73 [98]			1.63 [90]		

Tableau D.4: Erreurs[*#erreurs*] de l'algorithme tabou+diversification relativement à la meilleure solution connue: DIV4

s	1			(n-1)/2			(n-1)		
(n-m) f	0.25	0.50	0.75	0.25	0.50	0.75	0.25	0.50	0.75
(52-6)	0.71 [8]	0.53 [6]	1.24 [7]	1.70 [3]	0.48 [4]	0.27 [6]	0.88 [9]	0.77 [6]	0.34 [3]
(76-3)	0.99 [4]	0.72 [5]	1.31 [5]	0.98 [2]	1.73 [5]	0.64 [4]	2.30 [6]	0.86 [5]	0.35 [4]
(100-4)	0.54 [6]	0.72 [8]	1.24 [3]	0.58 [3]	1.46 [6]	0.72 [5]	0.33 [3]	0.69 [8]	0.69 [3]
(76-8)	2.28 [7]	1.13 [8]	0.87 [7]	3.64 [6]	0.35 [5]	0.69 [6]	3.68 [6]	0.93 [8]	0.64 [6]
(100-10)	0.19 [4]	0.66 [8]	0.68 [5]	1.87 [7]	0.57 [4]	0.69 [5]	2.12 [5]	0.35 [5]	0.67 [9]
Erreurs moyennes	1.14 [29]	0.75 [35]	1.07 [27]	1.75 [21]	0.92 [24]	0.60 [26]	1.86 [29]	0.72 [32]	0.54 [25]
Erreurs moyennes par *s*	0.99 [91]			1.09 [71]			1.04 [86]		

Tableau D.5: Erreurs[*#erreurs*] de l'algorithme tabou+diversification relativement à la meilleure solution connue: TAB

s / (n-m) / f	1			(n-1)/2			(n-1)		
	0.25	0.50	0.75	0.25	0.50	0.75	0.25	0.50	0.75
(52-6)	1.70 [10]	2.80 [8]	1.72 [7]	1.85 [7]	1.72 [7]	0.74 [6]	1.99 [8]	0.59 [7]	0.30 [4]
(76-3)	1.04 [6]	2.29 [9]	2.64 [8]	0.88 [8]	1.32 [7]	2.62 [7]	0.55 [7]	0.70 [6]	1.53 [7]
(100-4)	1.76 [9]	3.06 [10]	1.51 [6]	2.40 [9]	2.55 [8]	1.97 [9]	0.99 [8]	0.81 [10]	1.17 [9]
(76-8)	2.90 [7]	2.73 [9]	2.01 [10]	5.12 [8]	1.86 [10]	1.78 [9]	8.01 [10]	2.10 [10]	1.02 [8]
(100-10)	2.64 [10]	2.65 [9]	2.85 [8]	4.26 [8]	1.97 [7]	1.39 [7]	1.98 [5]	1.29 [9]	0.87 [8]
Erreurs moyennes	2.01 [42]	2.70 [45]	2.14 [39]	2.90 [40]	1.88 [39]	1.70 [38]	2.70 [38]	1.10 [42]	0.98 [36]
Erreurs moyennes par *s*	2.29 [126]			2.16 [117]			1.59 [116]		